SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-4022

The Parallel Decomposition of Linear Programs

by
Robert Entriken

TECHNICAL REPORT SOL 89-17

November 1989

Research and reproduction of this report were partially supported by the National Science Foundation grants DMS 8913089, DDM-8814253 and ECS-8715153; U.S. Department of Energy grants DE-FG03-87-ER-25028 and DE-FG03-87ER25030; the Office of Naval Research grant N00014-89-J-1659 and contract N00014-87-K-0142.

Abstract

This thesis introduces a new calculus for manipulating linear-program decomposition schemes. A linear program is represented by a *communication network*, which is decomposed by splitting nodes in two, and a transformation is defined to recover subproblems from the network. We also define a dual-symmetric oracle that provides solutions to linear programs, and can be performed by the simplex method, nested decomposition, and finally, parallel decomposition.

Two important classes of linear program serve as examples for the above calculus: staircase linear programs and stochastic linear programs. For the former case, a sophisticated yet experimental computer code has been written for an IBM 3090/600E with six processors. The code performs the parallel decomposition algorithm and is tested on twenty-two small to medium sized "real-world" problems. Experiments show that in addition to speedups provided by decomposition alone, performance is improved by using parallel processors.

(KR)

Acknowledgments

In the beginning Professor George Dantzig offered me support and guidance into the field of large-scale systems. I wanted to do something big, and with his encouragement I did. Patrick McAllister, Michael Saunders, and John Stone taught me how to handle the big computers and the large, capricious linear programs. They contributed the seed and early directions of growth. These early sprouts are now lofty branches. They were thickened by wonderful experiences at the Oak Ridge National Laboratory and the IBM T. J. Watson Research Center. At these places I gained access to the world of parallel computing, and the people there were always eager to explore promising new ideas.

Writing this thesis has taught me about coming to closure on a thought; sweating he details, focusing and converging. This would not have been possible without the ıelp of my readers: George Dantzig, Alan Hoffman, and Michael Saunders. Other priceless sounding boards were Chuck Romine at ORNL, and John Forrest, David Jensen, and Alan King at IBM. They cleared the fog and helped me to see the true structure of my thoughts.

Family is the fiber of our society that insures us against difficult times. I have felt at home with my colleagues and friends and relatives. At Stanford, I have the Operations Research Department, 661 Forrest, and Breakers Eating Club. In Philadelphia, I am proud to have the eternal backing of my parents, brothers and sister, and extended family.

For all this I am grateful.

R. E.

Palo Alto, California
August, 1989

Contents

List of Tables

List of Figures

Notation

Math Symbols

$\Re$ The ordered set of real numbers.

$\emptyset$ The empty set.

$\forall$ Means "for all".

$\in$ Means "is an element of".

$\supset$ Means "contains the set".

$\cap$ Intersection operation on sets.

$\cup$ Union operation on sets.

e A column vector of ones.

$(\cdot)^D$ The dual form of the linear program in the equation $(\cdot)$.

$\boxminus$ A binary operator that partitions the rows of a linear program.

$\boxplus$ A binary operator that partitions the columns of a linear program.

$\square$ The inverse of the row and column partition operators.

$\blacksquare$ End of proof.

$\square$ End of example.

Variables and Index Sets

All variables serve both as vectors in multidimensional real space and as sets of indices. The primal variables index the columns of the matrix A and row vector c^T, and the dual variables index the rows of A and the column vector b. The context will make clear whether a character such as x represents a real value or an index to a column.

Two types of variable are present in the formulation of a decomposition subproblem: original variables from the original problem and added variables for the purpose of appending and modifying the original ones.

The primal and dual variables are named with corresponding Roman and Greek characters. Even the functions of the characters as index sets and variables bear symmetric interpretations.

Index only

i, k Row indices.

j, l Column indices.

σ An index for the objective row.

s An index for the right-hand side.

Dual Variable or Row Index

λ An added dual variable on new constraints.

υ An added dual variable on the objective modification constraint.

π Original dual variables.

ψ An added dual variable on column accounting constraints.

ω An added dual variable on the right-hand side modification constraint.

θ An added dual variable on primal convexity constraints.

Primal Variable or Column Index

l An added primal variable to combine new columns.

u An added primal variable to implement a right-hand side modification.

x The original primal variables.

y An added primal variable to account passed primal solutions.

w An added primal variable to account the objective modification.

t An added primal variable on dual convexity constraints.

Variable only

α A non-negative scalar.

z An objective value.

Sets

$\mathcal{A}$ A closed polyhedral set representing a primal feasible region (in context).

$\mathcal{B}$ A closed polyhedral set representing a dual feasible region.

$\mathcal{C}$ A set containing column indices.

$\mathcal{G}^N$ The set containing all communication networks with N nodes.

$\dot{\mathcal{D}}$ A set containing dual extreme points.

$\vec{\mathcal{D}}$ A set containing dual extreme rays.

$\mathcal{R}$ A set containing row indices.

$\dot{\mathcal{P}}$ A set containing primal extreme points.

$\vec{\mathcal{P}}$ A set containing primal extreme rays.

Data Structures

Original Data

A Constraint coefficient matrix.

b Right-hand side vector.

c Vector of costs.

Added Data in Real Space

$\tilde{\Pi}$ An added data structure to contain extra constraints.

$\tilde{X}$ An added data structure to hold extra columns.

$\tilde{\psi}$ An added data structure for modifying an objective function.

$\tilde{y}$ An added data structure for modifying the right-hand side.

$\tilde{\theta}$ An added data structure containing the slope of the dual objective function in a dual extreme ray direction.

$\tilde{t}$ An added data structure containing the slope of the primal objective function in an extreme ray direction.

Added Data in Binary Space

$\tilde{\delta}$ A binary scalar indicating a dual extreme point in $\tilde{\psi}$.

$\tilde{d}$ A binary scalar indicating a primal extreme point in $\tilde{y}$.

$\tilde{\gamma}$ An binary vector indicating a corresponding dual extreme points in $\tilde{\Pi}$.

$\tilde{g}$ An binary vector indicating a corresponding primal extreme point in $\tilde{X}$.

I_a Subproblem interface matrix for arc a containing at most one unit entry per row and column.

Subscripts, Superscripts and Accents

The subscript n denotes information for node n, while the subscript a denotes information for arc a. Various math accents are used on both primal and dual variables throughout. The tilde accent, as in $\tilde{x}$, indicates a general solution value. The arrow accent, as in $\vec{x}$, indicates an extreme ray solution value. The dot accent, as in $\dot{x}$, indicates an extreme point solution value.

A_{ij} The ijth element of the matrix A.

b_i The ith element of column vector b.

c_j^T The jth element of row vector c.

$\tilde{x}_{ni}^k$ The ith element of the kth primal solution $\tilde{x}_n$ for node n.

$\tilde{\pi}_{nj}^k$ The jth element of kth dual solution $\tilde{\pi}_n$ for node n.

I_a^{ij} The ijth element of the matrix I_a

Dimensions

p The number of processors (in context).

N The number of subproblems.

K The number of times a given subproblem has been solved.

r_a The number of coupling rows between the subproblems connected by arc a.

r_n The number of non-zero rows in the column partition of subproblem n.

c_n The number of columns in the partition for subproblem n.

e_n The number of non-zero elements in the column partition of subproblem n.

$\bar{r}_n$ The number of rows in the formulation of subproblem n.

$\tilde{c}_n$ The number of columns in the formulation of subproblem n.

$\tilde{e}_n$ The number of non-zeros in the formulation of subproblem n.

$\hat{N}$ The maximum number of subproblems handled by the code.

$\hat{p}$ The maximum number of processors handled by the code.

$\hat{r}_n$ The maximum number of coupling constraints between adjacent subproblems that can be handled by the code.

Graph Theory

$\mathcal{N}$ The set of nodes in a graph.

n A node in $\mathcal{N}$.

$\mathcal{A}$ The set of arcs in a graph (in context).

a An arc in $\mathcal{A}$.

$\mathcal{T}_a$ The type for arc a (up, down, left, or right).

g A communication graph (when not subscripted).

h An incidence graph.

p A partition graph (in context).

$\mathcal{P}$ The set of all partition graphs (in context).

Problems and Solution Methods

NAME/n/p Problem NAME divided into n subproblems and solved using p processors.

ALG/p Algorithm ALG is run using p processors.

Multiple Meanings

The characters r, c, e, p, $\mathcal{A}$, and $\mathcal{P}$ can have multiple meanings. The first three are redefined in Chapter Four to refer to row, column and element dimensions of LPs. In Chapter Three, p is introduced as a partition graph, while in Chapter Four it refers to the number of computer processors applied to solving a test problem. Early in Chapter One, $\mathcal{A}$ refers to a primal feasible region, while later it is used as the set of arcs in a communication network. Finally, in Chapter One, $\mathcal{P}$, when accented with an arrow or dot, is a set of primal extreme rays or extreme points, but in Chapter Three, $\mathcal{P}$ is used exclusively as the set of all partition graphs in a communication network.

Chapter 1

Symbolic Decomposition

DESCRIBED herein is a methodology by which Linear Programs (LPs) can be decomposed into a collection of interdependent LPs and solved with a decomposition algorithm on a parallel computer. Equation (1.1) introduces the notation used throughout for linear program formulations:

$$\begin{array}{lll} \min\limits_{x \geq 0} & & c^T x = z \\ \text{s.t.} & \pi: & Ax \geq b. \end{array} \tag{1.1}$$

Corresponding to the constraints $Ax \geq b$ are dual variables π. The positioning of the dual variables to the left in (1.1) defines the correspondence between the slacks of the primal constraints and the dual variables. An analogous correspondence exists between the slacks of the dual constraints (reduced costs) and the primal variables.

Two important classes of problem will serve as guinea pigs to be dissected. Their anatomies are displayed in Figure 1.1. The dissection proceeds as a series of bisections or slices through the rows and columns of A, corresponding to a series of partitions of its row and column index sets. In the figure, the gray submatrices are where the nonzero coefficients are located, and the heavy lines with numerals 1, 2, 3, are the slices and the order in which they are made. Appendix B contains a collection of such nonzero coefficient patterns for a number of real-world staircase problems.

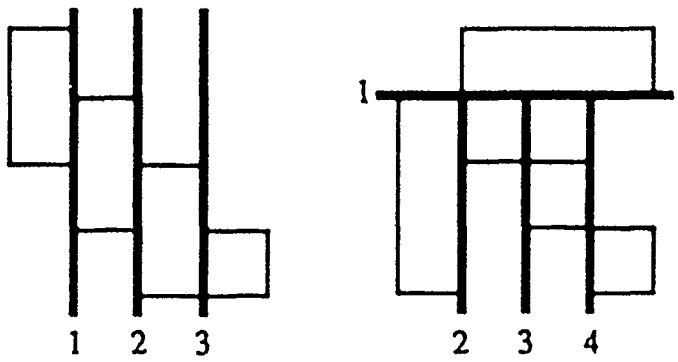

Figure 1.1: Partitioning variables and constraints.

The Staircase LP in Figure 1.1 has its *column index set C* partitioned into four sets by three slices. The first one slices off the leftmost block of nonzeros and the next two in turn slice off the remaining blocks in the same way. In a different manner, the *row index set* of the Two-Stage Stochastic LP is first partitioned into two sets, a top and a bottom; then the set associated with the bottom blocks of nonzeros is partitioned in a fashion similar to that of the Staircase LP. These two classes of linear program have many practical applications. There is an extensive literature on exploiting their special structure in order to develop an efficient solution algorithm; for example: [Dan59], [Zad62], [DGD64], [VS64], [Gla71], [Ho74], [DGS8].

The title of this thesis, *The Parallel Decomposition of Linear Programs*, means that these structures and others can be further exploited if the LP's are solved using parallel computers. The collection of interdependent LPs (subproblems) resulting from the decomposition prescribed above can be solved asynchronously on a parallel computer. Recalling that decomposition algorithms are iterative, we will show that the corresponding subproblems can be solved repeatedly, with information being passed from one to another until convergence is reached. Moreover, with a parallel computer we can solve these subproblems simultaneously with all processors efficiently employed, thus obtaining the overall solution more quickly.

The main contribution of this thesis involves developing a "symbolic calculus" for partitioning linear programs and demonstrating its usefulness on practical LP examples. Finally, we show that a parallel decomposition algorithm can indeed outperform serial algorithms, by experimenting with a computer code designed to solve staircase LPs.

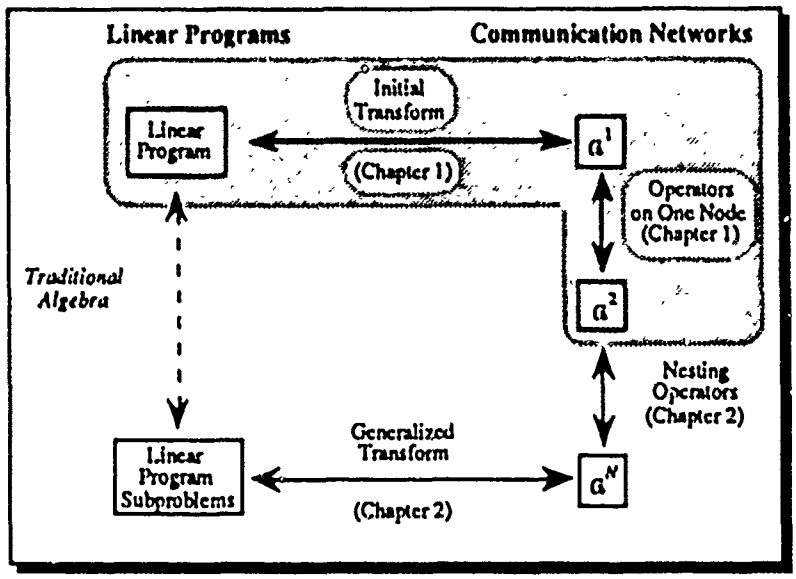

Figure 1.2: Symbolic Decomposition covered by Chapter One.

Figure 1.2 outlines the derivation of our symbolic decomposition calculus and its role in producing a system of subproblems. The left side of the diagram represents the traditional algebraic derivations of subproblem formulations. We propose a transform to a symbolic space that is based on network theory and we call communication networks. The symbol $\mathcal{G}^N$ represents the collection of all such networks on N nodes. In place of algebra, we define simple operators on the network that effect horizontal and vertical slices decomposition in ever more complex schemes. Finally, in Chapter 3 we provide a generalized parallel algorithm based on some given network in $\mathcal{G}^N$. This algorithm is a generalization of nested decomposition [Ho74, Abr83] and through

experiments on twenty-two staircase-LP test problems we show that decomposition algorithms can be sped up by parallel computers.

Given a dissection of the anatomy of a particular LP, like those in Figure 1.1, we can formulate certain well defined subproblems and a well defined algorithm to modify and solve the subproblems, thereby arriving at a solution to the original LP. We call the entire process *symbolic decomposition* because it is concerned not with actual data values but with the relationships between them (as necessary to solve the problem). The symbolic calculus we will describe allows for the possibility, if desired, of refining a dissection to the point where the individual blocks consist of single coefficients of the matrix A. Using this calculus, we can conveniently partition the blocks of a large-scale LP to exploit many different underlying patterns found in real-world problems.

Chapter One reviews the theory of decomposition by Goldman, Dantzig and Wolfe, and Benders, and introduces symbolic decomposition. It concludes with a theorem on subproblem interactions.

Decomposition, as described by Geoffrion, involves either some kind of restriction or some kind of relaxation of the original problem [Geo70]. Considerable advantage can be gained when the restriction or relaxation results in a much simpler problem. This is especially true when the original problem size is so large it would overwhelm the computer. The full problem can be broken into many smaller ones that can be solved to obtain an overall solution. This is decomposition.

All LP decomposition algorithms are based on two well known theorems. The first is the Goldman Resolution Theorem [Gol56], which states that a convex polyhedron can be described as a convex combination of its extreme points (provided such exist) plus a non-negative combination of its extreme rays (when not bounded). The second is that the solution of a linear program solved by the simplex method [Dan63] (whether primal or dual) is always at an extreme point (and/or an extreme ray).

There are two fundamental methods of decomposing a linear program into a collection of LP subproblems; the Dantzig-Wolfe method [DW61] and the Benders method

[Ben62]. They are the duals of each other. Using the former you slice horizontally, while with the latter you slice vertically.

The horizontal slice of the D-W method partitions the row indices $\mathcal{R}$ into two sets. We name them top and bottom. From them we generate two D-W subproblems. The top set corresponds to the traditional D-W Master problem, a relaxed version of (1.1) defined on only the constraints so indexed, while the bottom set corresponds to the D-W Slave problem. Information is passed up and down between them in the decomposition algorithm.

The dual method, that of Benders, operates via a partition of the *column index set* $\mathcal{C}$ into two sets: left and right. The left is used to generate the Master and the right the Slave.

We offer a caution on notation. The symbols for variables, i.e. x and π, are used in two ways that are context sensitive. In some places these symbols denote the values of primal and dual variables, but in other places they denote index sets for columns and/or rows of A, b, and c. Their proper interpretation should always be clear from their use.

1.1 Goldman's Resolution Theorem

Goldman's Resolution Theorem [Gol56] forms the basis for partially representing feasible regions of subproblems and generating sets of necessary conditions to describe them. The conditions are generated from successive solutions of the appropriate subproblems.

Let the closed polyhedral set $\mathcal{A} \doteq \{x : Ax \geq b,\ x \geq 0\}$, where A is a matrix of finite dimensions, and let the sets $\dot{\mathcal{P}}$ and $\vec{\mathcal{P}}$ consist of all the extreme points and rays, respectively, of $\mathcal{A}$, the primal feasible region of (1.1).

Theorem 1.1 (Goldman) *The set $\mathcal{A}$ can be expressed as a convex combination of its extreme points $\dot{\mathcal{P}}$ plus a non-negative combination of its extreme rays $\vec{\mathcal{P}}$:*

$$\mathcal{A} = \{x = \dot{x} + \alpha\vec{x}, \quad \forall \dot{x} \in conv(\dot{\mathcal{P}}), \vec{x} \in \vec{\mathcal{P}}, \alpha \geq 0\}.$$

In addition, the number of extreme points and rays will be finite.

The finiteness of a decomposition algorithm stems from the fact that the number of extreme points and extreme rays of the polyhedral set $\{x | Ax \geq b, x \geq 0\}$ is finite.

1.2 Solution Properties of Linear Programs

In order to enhance our geometric intuition of decomposition algorithms, we now describe the forms of information being passed between subproblems. First, let us define the sets $\dot{\mathcal{D}}$ and $\vec{\mathcal{D}}$ as the set of all extreme points and rays, respectively, of the set $\mathcal{B} = \{\pi : A^T\pi \leq c, \pi \geq 0\}$, the dual feasible region of (1.1). The simplex method and the following three theorems are due to Dantzig [Dan63].

Theorem 1.2 (Optimal Solution) *If an optimal solution to (1.1) exists, the simplex method will generate an optimal primal solution, $\dot{x} \in \dot{\mathcal{P}}$, and a vector of optimal dual multipliers, $\dot{\pi} \in \dot{\mathcal{D}}$. In addition, $c^Tx \geq b^T\dot{\pi}$, $\forall x \in \mathcal{A}$, with equality at $\dot{x}$.*

Corollary 1.3 (Separating Hyperplane (1)) *The hyperplane $\{x : c^Tx = b^T\dot{\pi}\}$ separates the set $\mathcal{A}$ from all points x that could give a lower value to c^Tx, where $\dot{\pi} \in \dot{\mathcal{D}}$ is a vector of optimal dual multipliers.*

Theorem 1.4 (Unbounded Solution) *If the solution to (1.1) is unbounded, the simplex method will give an extreme point, $\dot{x} \in \dot{\mathcal{P}}$, and an extreme ray, $\vec{x} \in \vec{\mathcal{P}}$, of $\mathcal{A}$ such that $c^T(\dot{x} + \alpha\vec{x}) \rightarrow -\infty$ as $\alpha \rightarrow \infty$. In addition, no feasible vector of dual multipliers π exists, so $\mathcal{B}$ is empty.*

Let $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ and $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ be corresponding row partitions. The problem (1.1) becomes

$$\begin{array}{lll} \min\limits_{x \geq 0} & & c^T x = z \\ \text{s.t.} & \pi_1 : & A_1 x \geq b_1 \\ & \pi_2 : & A_2 x \geq b_2. \end{array} \tag{1.2}$$

Define the top set $\mathcal{A}_1 = \{x : A_1 x \geq b_1\}$ and the bottom set $\mathcal{A}_2 = \{x : A_2 x \geq b_2, x \geq 0\}$, where the latter includes the non-negativity constraint. Note that their intersection is the original feasible region: $\mathcal{A} = \mathcal{A}_1 \cap \mathcal{A}_2$.

Theorem 1.5 (Infeasible Solution) *If there is no feasible solution for (1.2), the simplex method will find a vector of dual multipliers $(\vec{\pi}_1, \vec{\pi}_2) \in \vec{\mathcal{D}}$ that form an extreme ray of the polyhedron $\mathcal{B} = \{(\pi_1, \pi_2) : A_1^T \pi_1 + A_2^T \pi_2 \leq c, (\pi_1, \pi_2) \geq 0\}$. The ray satisfies $A_1^T \vec{\pi}_1 + A_2^T \vec{\pi}_2 \leq 0, (\vec{\pi}_1, \vec{\pi}_2) \geq 0$, and $b_1^T \vec{\pi}_1 + b_2^T \vec{\pi}_2 > 0$. If we assume that $\mathcal{A}_2 \neq \emptyset$ then $\vec{\pi}_1^T A_1 x < \vec{\pi}_1^T b_1 \ \forall x \in \mathcal{A}_2$.*

In the following corollary to Theorem 1.5, the dual ray $(\vec{\pi}_1 \quad \vec{\pi}_2)$ is identical to that in the theorem.

Corollary 1.6 (Separating Hyperplane (2)) *If both $\mathcal{A}_1$ and $\mathcal{A}_2$ are non-empty, the hyperplane $\{x : \vec{\pi}_1^T A_1 x = \vec{\pi}_1^T b_1\}$ strictly separates the sets $\mathcal{A}_1$ and $\mathcal{A}_2$, as does the hyperplane $\{x : \vec{\pi}_2^T A_2 x = \vec{\pi}_2^T b_2\}$.*

Table 1.1 summarizes the four combinations of primal and dual feasibility, and the results of the previous three theorems from the classical theory. For each combination, it lists the forms of the primal and dual solutions, with the primal forms handled by the simplex method in **bold face**—feasible optimal, feasible unbounded, and infeasible. In the primal infeasible cases, the dual ray is obtained at the end of Phase 1. Most algorithms terminate at this point without determining a dual extreme point when one exists.

Although the simplex method typically stops with only a dual ray when primal infeasible, it can yet obtain the non-bold face information. When a problem is known

Status	Solution Form	
Primal and Dual Optimal	**Primal Extreme Point**	**Dual Extreme Point**
Primal Unbounded and Dual Infeasible	**Primal Extreme Point & Ray**	*None*
Primal Infeasible and Dual Unbounded	*None*	**Dual Extreme Point & Ray**
Primal and Dual Infeasible	Primal Extreme Ray	**Dual Extreme Ray**

Table 1.1: Solutions of a primal formulation.

to be primal infeasible, it is easy to replace its right-hand side by one that makes it feasible. It will then finish either optimal or unbounded. If optimal, we have the "Primal Infeasible, Dual Feasible" case, and the optimal dual solution is the needed dual extreme point. If unbounded, we have the "Primal Infeasible, Dual Infeasible" case, and the ray associated with the unbounded solution is the missing primal extreme ray.

We now introduce the concept of an oracle. The word *oracle* usually refers to a magical source of truth. There is not much magic in our case, merely convenience. For the purpose of argument, the manner in which the oracle obtains information is not as important as the fact that it does provide it, and in a specific form. Our oracle will provide solutions to linear programs.

Definition 1.7 (An Oracle) *When consulted, an oracle $\mathcal{O}(\cdot)$ offers a "solution" for linear programs. In the case of (1.1), the oracle will generate either:*

1. *primal and dual optimal extreme points $\dot{x}$ and $\dot{\pi}$ satisfying $c^T\dot{x} = b^T\dot{\pi}$, or*
2. *a primal extreme ray $\bar{x}$ satisfying $c^T\bar{x} \leq 0$, or*

3. a dual extreme ray $\bar{\pi}$ satisfying $b^T\bar{\pi} \geq 0$.

In cases 2 and 3, we use a weak inequality to cover cases for which there is a ray of optimal solutions.

Lemma 1.8 *The Phase 1 / Phase 2 Simplex Method can perform $O(1.1)$.*

Proof: Compare the three oracle cases with Table 1.1. ∎

This oracle forms the basis for all of the following algorithmic results.

1.3 Dantzig-Wolfe Decomposition

We will now review Dantzig-Wolfe (D-W) decomposition [DW61] by partitioning the row index set of (1.2). We present decomposition algorithms as a combination of two parts: (a) the subproblem formulations, and (b) the protocol for passing information.

1.3.1 The D-W Subproblems

In a D-W decomposition scheme, let $\bar{X}_{\cdot j} \in \bar{\mathcal{P}}_2 \cup \bar{\mathcal{P}}_2$ be an extreme point or extreme ray of $\mathcal{A}_2 = \{x : A_2x \geq b_2, x \geq 0\}$. Let $\bar{g}_j = 1$ in the case of the former, and let $\bar{g}_j = 0$ in the latter. Let all such vectors $\bar{X}_{\cdot j}$ form the columns of a matrix $\bar{X}$. Then by Goldman's Theorem, any point $x \in \mathcal{A}_2$ can be represented by

$$\begin{aligned} x &= \bar{X}l, \quad l \geq 0 \\ 1 &= \bar{g}^T l, \end{aligned} \tag{1.3}$$

for some choice of variables l. The choice of l is not necessarily unique. Substituting the constraints on x from (1.3) for those corresponding to the region $\mathcal{A}_2$ in (1.2), we obtain the "Master" problem of the D-W decomposition scheme:

$$\begin{aligned} \min_{l\geq 0, x} \quad & c^Tx = z_1 \\ \text{s.t.} \quad \theta: \quad & \bar{g}^T l = 1 \\ \psi: \quad & \bar{X}l - Ix = 0 \\ \pi_1: \quad & A_1x \geq b_1. \end{aligned} \tag{1.4}$$

This system is of course equivalent to solving (1.2). On the surface it would appear that this transformation was made at the expense of greatly increasing the number of variables by including l. Suppose, however, that the oracle is consulted on a formulation of (1.4) where the columns of $\bar{X}$ contain only a subset of the extreme points and rays of $\mathcal{A}_2$. Let us assume that the oracle returns primal and dual extreme points $(\dot{l} \quad \dot{x})$ and $(\dot{\theta} \quad \dot{\psi} \quad \dot{\pi})$. We know that $\dot{x}$ must be in both $\mathcal{A}_2$ and $\mathcal{A}_1$, and that $\dot{\theta}\bar{g}^T + \dot{\psi}^T\bar{X} \leq 0$. If our present collection of extreme points and rays in $\bar{X}$ is sufficient to obtain a solution to (1.2), there can be no $x \in \mathcal{A}_2$ such that $\dot{\theta} + \dot{\psi}^T x > 0$. This can be determined by solving the "Slave" problem of the D-W decomposition scheme with $(\bar{\psi} \quad \bar{\delta} \quad \bar{\theta}) = (\dot{\psi} \quad 1 \quad \dot{\theta})$, defined as follows:

$$\begin{array}{lll} \min\limits_{x \geq 0,\, w} & & \bar{\delta} w = z_2 \\ \text{s.t.} & v: & \bar{\psi}^T x + \bar{\delta} w \geq -\bar{\theta} \\ & \pi_2: & A_2 x \qquad \geq b_2. \end{array} \tag{1.5}$$

The motivation for this problem is to answer the question:

Is there a point $x \in \mathcal{A}_2$ such that $\dot{\theta} + \dot{\psi}^T x > 0$?

For a dual feasible solution to (1.5) v will equal 1, meaning that its corresponding constraint is binding. In which case, $w = -\bar{\psi}^T x - \bar{\theta}$ and we are minimizing w over all $x \in \mathcal{A}_2$. Therefore, if $z_2 \geq 0$, there can be no $x \in \mathcal{A}_2$ such that $\dot{\theta} + \dot{\psi}^T x > 0$, and in answer to the above question: there is no such point. Further, there are no extreme points or rays of $\mathcal{A}_2$ which if added to our present collection in $\bar{X}$ could improve the overall solution. We have a solution to (1.1).

In (1.5), $(\bar{\psi}, \bar{\delta}, \bar{\theta})$ is an oracle-provided extreme point or extreme ray of the dual feasible region of (1.4) for some $(\dot{\mathcal{P}}_2, \vec{\mathcal{P}}_2)$. If it is a dual extreme point, $(\bar{\psi}, \bar{\delta}, \bar{\theta}) = (\dot{\psi}, 1, \dot{\theta})$, and if it is a dual extreme ray, $(\bar{\psi}, \bar{\delta}, \bar{\theta}) = (\vec{\psi}, 0, \vec{\theta})$.

When $\bar{\delta}$ equals zero in the extreme ray case, (1.5) will have a vacuous objective and becomes a feasibility problem. We need only find a feasible point to solve it. The next section details the D-W method of solution, in which we will see that having

$\tilde{\delta} = 0$ directs the Slave to find points in $\mathcal{A}_2$ that could make an infeasible Master feasible.

Note in the Master that if there are no extreme points among the columns of $\tilde{X}$ then $\tilde{g} = 0$ and (1.4) is infeasible via $0 \cdot l = 1$. To ensure that $\mathcal{A}_2$ contains at least one extreme point we make x non-negative in $\mathcal{A}_2$ rather than in $\mathcal{A}_1$.

Equation (1.4) is commonly referred to as the *Master* problem because its dual solutions $\tilde{\psi}$ impact the objective of (1.5), the *Slave*, to select extreme information from the set $\mathcal{A}_2$ that will lead to an overall optimum solution of (1.2). In the following chapters, the Master/Slave distinction is not sufficient when the dual of this algorithm is incorporated. For this reason, all such LPs will be referred to as subproblems (being subordinate to the original problem), and further, (1.4) will be referred to as the *top* subproblem, and (1.5) as the *bottom* subproblem.

1.3.2 The Dantzig-Wolfe Method

In the Dantzig-Wolfe method, the top subproblem (1.4) is solved with $\dot{\mathcal{P}}_2$ and $\vec{\mathcal{P}}_2$ restricted to promising subsets of the extreme points and extreme rays of $\mathcal{A}_2$. Initially those subsets are empty and we need to build them up to the point where they are sufficient for determining the solution to the original problem (1.2). On each major iteration between solving (1.4) and (1.5), one of the sets is expanded: $\dot{\mathcal{P}}$ if (1.5) is optimal, or $\vec{\mathcal{P}}$ if (1.5) is unbounded. In (1.4) the values of x are restricted to be convex combinations of the points in $\dot{\mathcal{P}}_2$ and non-negative linear combinations of the rays in $\vec{\mathcal{P}}_2$.

In the spirit of Theorem 1.1, the constraints associated with the dual variables θ and ψ in (1.4), along with $l \geq 0$, form a *partial representation* of $\mathcal{A}_2$. Consistent with our earlier definitions, we define this set as

$$\tilde{\mathcal{A}}_2 = \{x : x = \tilde{X}l,\ \tilde{g}^T l = 1,\ l \geq 0\}$$

and so (1.4) becomes

$$\text{minimize } c^T x,\ x \in \mathcal{A}_1 \cap \tilde{\mathcal{A}}_2.$$

We pictorially represent the D-W decomposition of (1.2) in Figure 1.3. On the left is an anatomic representation of the row index partition, and on the right is a 2-dimensional geometric representation of the intersection of the two polyhedral sets $\mathcal{A}_1$ and $\mathcal{A}_2$.

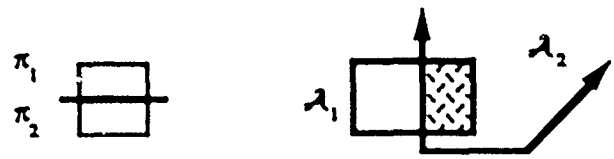

Figure 1.3: Partitioning constraints and set intersection.

We graphically and geometrically represent the D-W algorithm with its top and bottom subproblems (1.4) and (1.5) in Figure 1.4. On the left, the two circles represent the subproblems, and the arrows, or arcs, represent channels of communication for their solution information. This diagram will be referred to as a *communication network*, on which the decomposition algorithm bases its protocol for passing messages. The arcs in the diagram are of two types: *up* and *down*. Up arcs always pass primal solutions that are collected at the destinations and used to form partial representations of the primal feasible regions of the sources. A down arc always passes dual solutions, of which only the most recent is retained at the destination and used to modify the objective function of that subproblem.

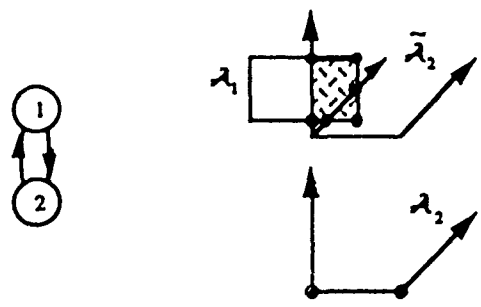

Figure 1.4: Subproblem communication and partial representation of $\mathcal{A}_2$.

On the right of Figure 1.4 are 2-dimensional geometric representations of the feasible regions of (1.4) and (1.5). The two dots in the corners of the region $\mathcal{A}_2$ are the totality of its extreme points, some of which are passed to (1.4). The region $\tilde{\mathcal{A}}_2$ in the drawing above $\mathcal{A}_2$ is based on any combination of extreme points and rays passed. One of the rays is drawn twice to show its dependence on the extreme points passed. The six dots in the intersection of $\mathcal{A}_1$ and $\tilde{\mathcal{A}}_2$ are the possible extreme point solutions to (1.4), depending on which combination of the extreme points and rays of $\mathcal{A}_2$ were used to construct $\tilde{\mathcal{A}}_2$.

Given that (1.4) is initially infeasible, the first order of business is to find a point in $\mathcal{A}_1 \cap \mathcal{A}_2$ in order to demonstrate the feasibility of (1.2), then to find a feasible point that minimizes the objective function. The next theorem describes an algorithm that accomplishes these tasks. The set $\tilde{\mathcal{A}}_2 = \{x \mid \dot{x} + \alpha\vec{x}, \forall\, \dot{x} \in conv(\dot{\mathcal{P}}_2)$ and $\vec{x} \in \vec{\mathcal{P}}_2\}$ begins empty and is augmented in each cycle between Steps 2 and 3. It in turn defines the added data $\bar{g}$ and $\bar{X}$ in the formulation of (1.4).

Theorem 1.9 (Dantzig-Wolfe Method) *This procedure performs $\mathcal{O}$(1.2):*

1. *Let $\dot{\mathcal{P}}_2 = \vec{\mathcal{P}}_2 = \emptyset$.*
2. *Consult $\mathcal{O}$(1.4) and if it returns*
   - *an optimal dual extreme point, let $(\bar{\psi}, \bar{\delta}, \bar{\theta}) \leftarrow (\dot{\psi}, 1, \dot{\theta})$;*
   - *a primal extreme ray,* STOP*—$\mathcal{O}$(1.2) is $\vec{x}$;*
   - *a dual extreme ray, let $(\bar{\psi}, \bar{\delta}, \bar{\theta}) \leftarrow (\vec{\psi}, 0, \vec{\theta})$.*
3. *Consult $\mathcal{O}$(1.5) and if it returns*
   - *an optimal primal extreme point, and*
     i) $\dot{z}_2 < 0$, let $\dot{\mathcal{P}}_2 \leftarrow \dot{\mathcal{P}}_2 \cup \{\dot{x}\}$ and go to Step 2;
     ii) $\dot{z}_2 \geq 0$, STOP*—if $\bar{\delta} = 1$, $\mathcal{O}$(1.2) is $\dot{x}$ from $\mathcal{O}$(1.4) and $(\dot{\pi}_1^T \quad \dot{\pi}_2^T)$, else $\mathcal{O}$(1.2) is $(\vec{\pi}_1^T \quad \dot{\pi}_2^T)$;*
   - *a primal extreme ray, let $\vec{\mathcal{P}}_2 \leftarrow \vec{\mathcal{P}}_2 \cup \{\vec{x}\}$ and go to Step 2;*
   - *a dual extreme ray,* STOP*—$\mathcal{O}$(1.2) is $(0 \quad \vec{\pi}_2^T)$.*

Proof: We will work from four cases and then show finiteness.

Case 1: If $\mathcal{A}_1 = \emptyset$ and $\mathcal{A}_2 \neq \emptyset$, $\mathcal{O}(1.4)$ will finish infeasible, implying $z_1 = \bar{\pi}_1^T(b_1 - A_1 x) > 0 \quad \forall x \in \mathcal{A}_2$. Then, $\mathcal{O}(1.5)$ will return optimal extreme points with $z_2 > 0$. $\mathcal{O}(1.2)$ returns $(\bar{\pi}_1 \quad \dot{\pi}_2)$.

Case 2: If $\mathcal{A}_1 \neq \emptyset$ and $\mathcal{A}_2 = \emptyset$, $\mathcal{O}(1.5)$ will return a dual ray. $\mathcal{O}(1.2)$ returns $(0 \quad \bar{\pi}_2^T)$.

Case 3: If $\mathcal{A}_2 \neq \emptyset$ and $\mathcal{A}_1 \neq \emptyset$ and in Step 2, $\mathcal{O}(1.4)$ is a dual ray, then by Theorem 1.5, the hyperplane $\{x : \bar{\theta} + \bar{\psi}x = 0\}$ strictly separates $\mathcal{A}_1$ and $\tilde{\mathcal{A}}_2$. According to Step 3 and Equation (1.5) we must find a point as far as possible on the opposite side of this hyperplane from $\tilde{\mathcal{A}}_2$ that also lies in $\mathcal{A}_2$. If no such point exists, $\dot{z}_2 = 0$ and the original problem (1.2) must be infeasible; $\mathcal{O}(1.2)$ returns $(\bar{\pi}_1^T \quad \dot{\pi}_2^T)$. On the other hand, if one does exist, go back to Step 2.

Case 4: If $\mathcal{A}_2 \neq \emptyset$ and $\mathcal{A}_1 \neq \emptyset$ and in Step 2, $\mathcal{O}(1.4)$ is a dual point, then by Theorem 1.2, the hyperplane $\{x : \bar{\theta} + \bar{\psi}x = 0\}$ separates $\tilde{\mathcal{A}}_2$ from all points $x \in \mathcal{A}_2$ that could give a better value of $c^T x$. According to Step 3 and Equation (1.5) we must find a point as far as possible on the opposite side of this hyperplane from $\tilde{\mathcal{A}}_2$ that also lies in $\mathcal{A}_2$. If no such point exists, the original problem (1.2) must be optimal, and $\mathcal{O}(1.2)$ returns x, $(\dot{\pi}_1^T \quad \dot{\pi}_2^T)$, where x is the present primal optimal solution to (1.4). On the other hand, if one does exist, go back to Step 2.

Finiteness: Step 3 can never pass the same information twice because any dual solution from (1.4) satisfies $\bar{\theta} + \bar{\psi}x \geq 0$ for all $x \in \tilde{\mathcal{A}}_2$, and the procedure would stop either optimal or infeasible. The procedure is finite because it is drawing upon a finite set of extreme-point and extreme-ray data of $\mathcal{A}_2$ that can be passed from (1.5) to (1.4). At any point in the algorithm some subset of this information is in the top subproblem. Each such subset must be different because the top's feasible region is expanded in each cycle. Since the number of subsets is finite and none is repeated, the algorithm must eventually stop. ∎

1.3.3 The D-W Communication Network

The goal of this chapter and the next is to characterize the space of all decomposition schemes by deriving the subproblem formulations in increasingly complex steps. At each step, the system of subproblems will be transformed into an equivalent symbolic representation, and a symbolic operation will be defined to mimic the decomposition step.

To introduce symbolic decomposition and to develop a formal representation of D-W decomposition, let us formalize the concept of a *communication network*. It is a symbolic representation of a decomposition scheme containing the information necessary to define all of the subproblems, symbolized by nodes, and their interactions, symbolized by arcs.

Let $\mathcal{G}^N$ be the set of all communication networks on N nodes. These networks are alternative representations of LP decomposition schemes. We will define a transformation from the space of all decomposition schemes to all communication networks. It will be shown that this transformation is reversible.

Definition 1.10 (Communication Network) *A communication network is a five-tuple. For example,*

$$g = (\mathcal{N}, \mathcal{R}_n, \mathcal{C}_n, \mathcal{A}, \mathcal{T}_a), \quad \forall n \in \mathcal{N}, a \in \mathcal{A},$$

where the tuples are defined to be

$\mathcal{N}$ *set of nodes,* $\mathcal{N} \neq \emptyset$,

$\mathcal{R}_n$ *node n's row index set,*

$\mathcal{C}_n$ *node n's column index set,*

$\mathcal{A}$ *set of arcs,* $\mathcal{A} = \{(n_1, n_2) \in \mathcal{N}^2 :$ *there is an arc from node* n_1 *to* $n_2\}$, *and*

$\mathcal{T}_a$ *the type for arc a (up, down, left, or right), where if* $\mathcal{A} = \emptyset$ *there are no* $\mathcal{T}_a$.

The arc types left and right are used in the Benders decomposition method and explained later. They are included here for completeness of the definition.

The simplest network is one with only one node and no arcs. It symbolizes a linear program that has not been decomposed. We offer g_1 as an example:

$$g_1 = (\{0\}, \pi, x, \emptyset) \in \mathcal{G}^1.$$

The node is numbered zero. The sets of row and column indices are respectively π and x. The set of arcs is empty, and the arc types are not applicable. The network g_1 symbolizes everything in the formulation of the problem (1.1) except the actual values of the data (A, b, c). A communication network, together with the data (A, b, c), is sufficient information to obtain a solution to (1.1).

Subproblem index sets are used in the definition of the forward transformation, which was first depicted in the shaded region of Figure 1.2. An overbar distinguishes them from the node index sets of communication networks. Typically, $\bar{\mathcal{R}}_n \supseteq \mathcal{R}_n$ and $\bar{\mathcal{C}}_n \supseteq \mathcal{C}_n$.

Definition 1.11 (Subproblem Index Sets) *Let the set $\bar{\mathcal{R}}_n$ contain all row indices of the linear program subproblem associated with node n, and let $\bar{\mathcal{C}}_n$ contain all column indices of the linear program subproblem associated with node n.*

Thus in the example above with one node and no arcs, $\bar{\mathcal{R}}_0 = \pi$, i.e., all rows, and $\bar{\mathcal{C}}_0 = x$, i.e., all columns.

We now define a communication network based on the Dantzig-Wolfe (top and bottom) subproblems. This operation was referred to when explaining Figure 1.2. We are taking the initial step from linear programs to communication networks.

Definition 1.12 (Forward Transform) *The forward transform from a D-W decomposition scheme to the communication network g_D is a five-step process:*

1. *Define the set $\mathcal{N}$ having one node for each subproblem.*
2. *For each node $n \in \mathcal{N}$, define the elements of the row index set $\mathcal{R}_n = \bar{\mathcal{R}}_n \cap \mathcal{R}$.*
3. *For each node $n \in \mathcal{N}$, define the elements of the column index set $\mathcal{C}_n = \bar{\mathcal{C}}_n \cap \mathcal{C}$.*

4. *For each subproblem $n_1 \in \mathcal{N}$ containing constraints indexed by 0 and ψ that receive information from another subproblem $n_2 \in \mathcal{N}$, define an arc $(n_2\, n_1) \in \mathcal{A}$ of type $\mathcal{T}_{n_2 n_1} = up$.*

5. *For each subproblem $n_1 \in \mathcal{N}$ containing constraints indexed by v that receive information from another subproblem $n_2 \in \mathcal{N}$, define an arc $(n_2\, n_1) \in \mathcal{A}$ of type $\mathcal{T}_{n_2 n_1} = down$.*

With these transformation rules, we can derive the network in Figure 1.4 corresponding to the two Dantzig-Wolfe subproblems (1.4) and (1.5).

1. $\mathcal{N} = \{1, 2\}$ since there are two subproblems, with 1 corresponding to the top and 2 corresponding to the bottom.

2. $\mathcal{R}_1 = \pi_1$ and $\mathcal{R}_2 = \pi_2$ signifying that the rows are partitioned so that those associated with π_1 go to the top subproblem and those associated with π_2 go to the bottom subproblem.

3. $\mathcal{C}_1 = \mathcal{C}_2 = x$ as the columns were not partitioned (that comes later).

4. 0 and ψ appear in (1.4) so let $\mathcal{A} \leftarrow \mathcal{A} \cup \{(21)\}$ and $\mathcal{T}_{21} = \text{up}$.

5. v appears in (1.5) so let $\mathcal{A} \leftarrow \mathcal{A} \cup \{(12)\}$ and $\mathcal{T}_{12} = \text{down}$.

In summary,

$$g_D = (\{1, 2\}, \pi_1, \pi_2, x, x, \{(12), (21)\}, \text{down, up}) \in \mathcal{G}^2.$$

In this notation the set of nodes appears first, followed by a list of row index sets, one for each node, followed by a corresponding list of column index sets. Next is the set of arcs followed by a list of arc types.

Theorem 1.13 (The Reverse Transform) *The reverse transform from g_D back to a D-W decomposition scheme proceeds as follows:*

1. *For each node n, create a subproblem beginning with the original constraints of the form*

$$A_i x \geq b_i, \quad \forall i \in \mathcal{R}_n.$$

2. *Corresponding to the up arc from 2 to 1, include constraints in the top subproblem of the form*

$$\bar{g}^T l = 1, \quad \bar{X} l = x, \quad l \geq 0,$$

where $\bar{g}$ and $\bar{X}$ are information transported by the up arc.

3. *Corresponding to the down arc from 2 to 1, include constraints in the bottom subproblem of the form*

$$\bar{\delta} w \geq -\bar{\theta} - \bar{\psi} x,$$

where $(\bar{\theta}, \bar{\delta}, \bar{\psi})$ is information transported by the down arc. In addition, place the term $+\bar{\delta} w$ in the objective row.

4. *Place the term $+c^T x$ in the objective row of the top subproblem.*

5. *Place the non-negativity constraints $x \geq 0$ in the bottom subproblem.*

Proof: Proof by example (WLOG). ∎

We use the node index set $\mathcal{R}_0$ in the following definition.

Definition 1.14 (Dantzig-Wolfe Operator, $\boxminus$) *The Dantzig-Wolfe operator $\boxminus$ maps $\mathcal{G}^1$ into $\mathcal{G}^2$ using a partition $[P_1, P_2]$ of $\mathcal{R}_0$:*

$$\mathcal{G}^1 \boxminus [P_1, P_2] \rightarrow \mathcal{G}^2.$$

In words this means:

Apply Dantzig-Wolfe decomposition to the linear program associated with node n in the communication network. Partition the constraints so that

those corresponding to the dual variables in P_1 are in the top (D-W Master) subproblem and those corresponding to the dual variables in P_2 are in the bottom (D-W Slave) subproblem. The algorithm described in Theorem 1.9 is implicitly associated with the resultant communication network.

This is our most elementary row partition. Begin with the network g_1, which has only one node, and map it into a two-node network corresponding to the left side of Figure 1.4. Thus, $g_1 = (\{0\}, \pi_1 \cup \pi_2, x, \emptyset)$ and

$$g_D = g_1 \boxminus [\pi_1, \pi_2],$$

where g_D was given above. From g_D we can determine which node is the top or bottom by the arcs that link them. The up arc must be destined for the top node (subproblem).

The next definition is needed to establish an equivalence between subproblem schemes and communication networks. We define the Inverse Operator on only those collections of nodes that, once collapsed, can be re-split to regain the original network. Later we will rely on this reversibility property of the inverse operator when characterizing the space of all communication networks, $\mathcal{G}^N$.

Definition 1.15 (Inverse Operator, $\square$) *The inverse operator $\square$ takes a set of nodes and collapses it back into a single node. It is defined such that*

$$g_1 = (g_1 \boxminus [P_1, P_2]) \square \{1, 2\},$$

for all partitions $[P_1, P_2]$ of $\mathcal{R}_0$.

In the expression $g_1 = g_D \square \{1, 2\}$, two nodes are collapsed into one, and the arcs are discarded.

For networks with more than two nodes, the inverse operator can be thought of as identifying *implicit* subproblems, i.e., collections of subproblems that imitate, in concert, a subproblem that does not exist explicitly. This is explained more carefully in Chapter Two.

The implication of the triple (the transform, the D-W operator, and the inverse operator) is that we have created the symbolic space $\mathcal{G}^2$ within which we can mimic the algebraic manipulations of decomposition. By partitioning the index set of a node in a specific way, we mimic the creation of two subproblems from a single LP. Alternatively, the original LP can be regained by combining the index sets of the two nodes, also in a specific way.

1.4 Benders Decomposition

The purpose of this section is to derive an oracle for LPs that have been sliced vertically (partitioned by columns) by dualizing the concepts we have discussed for those sliced horizontally (partitioned by rows). The theorems and definitions of the previous section will thus return in their dual forms. To characterize $\mathcal{G}^1$ and $\mathcal{G}^2$, we first complete the forward transform by including vertical slicing and deriving its companion oracle. Then we define in sequence: the dual operator, its inverse, and the dual network.

1.4.1 The Subproblems

Now consider partitioning the LP (1.1) where $A = (A_3 \quad A_4)$ and $c^T = (c_1^T \quad c_2^T)$, and $g_1 = (\{0\}, \pi, x_1 \cup x_2, \emptyset)$. The index sets x_1 and x_2 form a partition of the column index set x of (1.1) and π is its row index set. With this column partition the LP problem becomes

$$\begin{array}{lll} \min\limits_{\substack{x_1 \geq 0 \\ x_2 \geq 0}} & & c_1^T x_1 + c_2^T x_2 = z \\ \text{s.t.} & \pi: & A_3 x_1 + A_4 x_2 \geq b. \end{array} \tag{1.6}$$

The subproblems and the oracle for Benders decomposition [Ben62] can be derived directly from D-W decomposition by replacing primal/dual steps by corresponding dual/primal steps. We want the decomposition-style oracle $\mathcal{O}(1.6)$ that utilizes $\mathcal{O}(1.7)$ and $\mathcal{O}(1.8)$. A vertical slice through A between x_1 and x_2 partitions its column index

set C. To derive the Benders subproblems, we take the dual of (1.6), apply the D-W operator, and then take the duals once again of the resulting subproblems.

Taking the dual of (1.6) gives

$$\begin{array}{lll} \max\limits_{\pi \geq 0} & & b^T\pi = z \\ \text{s.t.} & x_1: & A_3^T\pi \leq c_1 \\ & x_2: & A_4^T\pi \leq c_2, \end{array} \qquad (1.6)^D$$

where we have introduced the notation $(\cdot)^D$ to indicate the dual of the LP argument.

Applying D-W decomposition to this problem as was done for (1.2) gives us the two subproblem formulations $(1.7)^D$ and $(1.8)^D$. We have substituted the variables (t, y, λ, u, w) for $(\theta, \psi, l, v, \omega)$, and the data $(\tilde{\gamma}, \tilde{\Pi}, \tilde{y}, \tilde{d}, \tilde{t})$ for $(\tilde{g}, \tilde{X}, \tilde{\psi}, \tilde{\delta}, \tilde{\theta})$. They are corresponding Greek and Roman characters. Specifically,

$$\begin{array}{lll} \max\limits_{\pi, \lambda \geq 0} & & b^T\pi = z_1 \\ \text{s.t.} & t: & \tilde{\gamma}^T\lambda = 1 \\ & y: & \tilde{\Pi}\lambda - I\pi = 0 \\ & x_1: & A_3^T\pi \leq c_1, \end{array} \qquad (1.7)^D$$

and

$$\begin{array}{lll} \max\limits_{\pi \geq 0, \omega} & & \tilde{d}\omega = z_2 \\ \text{s.t.} & u: & \tilde{y}^T\pi + \tilde{d}\omega \geq -\tilde{t} \\ & x_2: & A_4^T\pi \leq c_2, \end{array} \qquad (1.8)^D$$

Define $\mathcal{B}_1 = \{\pi : A_3^T\pi \leq c_1\}$ and $\mathcal{B}_2 = \{\pi : A_4^T\pi \leq c_2, \pi \geq 0\}$. Note that if $\mathcal{B}$ is the feasible region of $(1.6)^D$ then $\mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2$, and the columns of the matrix $\tilde{\Pi}$ contain extreme points and extreme rays of $\mathcal{B}_2$. We define $\dot{\mathcal{D}}_2$ and $\vec{\mathcal{D}}_2$ to be the respective subsets of the extreme points and rays of $\mathcal{B}_2$ and get $\tilde{\Pi}_{\cdot j} \in \dot{\mathcal{D}}_2 \cup \vec{\mathcal{D}}_2$. Since we define $\tilde{\gamma}_j$ to be 1 if $\tilde{\Pi}_{\cdot j} \in \dot{\mathcal{D}}_2$ and 0 otherwise, the π in $(1.7)^D$ must be an element of $\mathcal{B}$ if $\dot{\mathcal{D}}_2$ and $\vec{\mathcal{D}}_2$ contain all of the extreme points and extreme rays in $\mathcal{B}_2$.

The respective duals of the previous two maximization problems give us the standard subproblems of Benders decomposition:

$$\begin{array}{lll} \min\limits_{x_1 \geq 0, y, t} & c_1^T x_1 & + \ t = z_1 \\ \text{s.t.} \quad \pi: & A_3 x_1 - Iy & = b \\ \lambda: & \bar{\Pi}^T y + \bar{\gamma} t & \geq 0, \end{array} \tag{1.7}$$

and

$$\begin{array}{lll} \min\limits_{\substack{x_2 \geq 0 \\ u \geq 0}} & c_2^T x_2 - \bar{t} u = z_2 & \\ \text{s.t.} \quad \omega: & \bar{d} u = \bar{d} & \\ \pi: & A_4 x_2 + \bar{y} u \geq 0. & \end{array} \tag{1.8}$$

1.4.2 The Benders Method

To construct a decomposition procedure that performs $\mathcal{O}(1.6)$, we first make the following definition.

Definition 1.16 (Dual Oracle) *The dual of an oracle, symbolized as $\mathcal{O}^D(\cdot)$, interprets the dual solutions of $\mathcal{O}(\cdot)$ as primal solutions, and the primal solutions of $\mathcal{O}(\cdot)$ as dual solutions.*

As defined, the dual of the dual oracle is the original oracle. We get the following property by combining the dual oracle with the dual of a linear program.

Property 1.17 (Oracle Dual Symmetry) *The oracle $\mathcal{O}(\cdot)$ is dual symmetric in that*

$$\mathcal{O}^D(\cdot)^D = \mathcal{O}(\cdot).$$

We are reusing our notation $(\cdot)^D$ to indicate the dual of the LP argument. To verify this property, consult Table 1.1 and note that the simplex method can perform $\mathcal{O}(\cdot)^D$.

Corollary 1.18 (Benders Method) *The following procedure can perform $\mathcal{O}(1.6)$:*

1. *Let $\dot{\mathcal{D}} = \vec{\mathcal{D}} = \emptyset$.*
2. *Consult $\mathcal{O}(1.7)$ and when it returns*

- *an optimal primal extreme point, let* $(\bar{y}, \bar{d}, \bar{t}) \leftarrow (\dot{y}, 1, \dot{t})$;
- *a dual extreme ray,* STOP—$\mathcal{O}(1.6)$ *is* $\vec{\pi}$;
- *a primal extreme ray, let* $(\bar{y}, \bar{d}, \bar{t}) \leftarrow (\vec{y}, 0, \vec{t})$.

3. *Consult* $\mathcal{O}(1.8)$ *and when it returns*
   - *an optimal dual extreme point and*
     - *i)* $z_2 < 0$, *let* $\dot{\mathcal{D}}_2 \leftarrow \dot{\mathcal{D}}_2 \cup \{\dot{\pi}\}$, *and go to Step 2;*
     - *ii)* $z_2 \geq 0$, STOP—*if* $\bar{\delta} = 1$, $\mathcal{O}(1.6)$ *is* $(\dot{x}_1^T \quad \dot{x}_2^T)$ *and* $\dot{\pi}$, *the latter coming from* $\mathcal{O}(1.7)$, *otherwise* $\mathcal{O}(1.6)$ *is* $(\vec{x}_1^T \quad \dot{x}_2^T)$;
   - *a dual extreme ray, let* $\vec{\mathcal{D}}_2 \leftarrow \vec{\mathcal{D}}_2 \cup \{\vec{\pi}\}$, *and go to Step 2;*
   - *a primal extreme ray,* STOP—$\mathcal{O}(1.6)$ *is* $(0 \quad \vec{x}_2^T)$.

Proof: Begin with the Dantzig-Wolfe Oracle in Theorem 1.9, and wherever $\mathcal{O}(\cdot)$ is consulted, replace that consultation by $\mathcal{O}^D(\cdot)^D$, using the Oracle Dual Symmetry Property. Next, for each of the three cases for solutions, interchange the words *primal* and *dual*, and replace the oracle consultations as before, but with $\mathcal{O}(\cdot)^D$, using the definition of a dual oracle. Finally, replace the subproblem formulations with their duals, and replace the oracle consultations as before but with $\mathcal{O}(\cdot)$, using the definition of $(\cdot)^D$. The oracles $\mathcal{O}(1.7)$ and $\mathcal{O}(1.8)$ for the Benders left and right subproblems then become the results we require to complete $\mathcal{O}(1.6)$. ∎

As a note on the milestones of the procedure above, once having found $\dot{\pi} \in B_1 \cap B_2$, we have demonstrated primal boundedness for (1.6). To continue, we must work toward dual optimality in order to show primal optimality. If we find dual unboundedness then the primal form (1.6) is infeasible.

We will anatomically and graphically represent the Benders decomposition of (1.6) as Figure 1.5. On the left, the matrix A is sliced vertically between x_1 and x_2 to symbolize the partition of the set x into the sets x_1 and x_2. On the right is the communication network, on which the algorithm bases its protocol for passing information and making modifications. The right arc always passes primal solutions, of

which only the most recent is retained at the destination (node 2) and used to modify the right-hand side of the subproblem. The left arc always passes dual solutions that are collected at the destination (node 1) and used to form a partial representation of the dual feasible region of the source node.

Figure 1.5: Partitioning constraints and communication.

1.4.3 Dual Communication Network Theory

The network theory corresponding to the above subproblems and oracle derivations is itself replete with the use of duality concepts. First, the Forward Transform (from subproblems to networks) needs two new rules that are dual to Rules 4 and 5 and serve to transform the Benders (left and right) subproblems. Next, we present the dual to the D-W network, which is generated by the Benders operator.

Definition 1.19 (Forward Transform continued) *The forward transform from a Benders decomposition scheme is a five-step process. The first three steps are taken as those in the prior Forward Transform definition, and the last two are additions to the prior that complete the definition over $\mathcal{G}^1$ and $\mathcal{G}^2$.*

6. *For each subproblem $n_1 \in \mathcal{N}$ containing variables named y and t, and for every other subproblem $n_2 \in \mathcal{N}$ that provides information for those columns, define an arc $(n_2\, n_1) \in \mathcal{A}$ of type $\mathcal{T}_{n_2 n_1} = left$.*

7. *For each subproblem $n_1 \in \mathcal{N}$ containing variables named u, and for every other subproblem $n_2 \in \mathcal{N}$ that provides information for those columns, define an arc $(n_2\, n_1) \in \mathcal{A}$ of type $\mathcal{T}_{n_2 n_1} = right$.*

Theorem 1.20 (Dual Reverse Transform) *The reverse transform back to a Benders decomposition scheme from g_B proceeds as follows:*

1. *For each node n, create a subproblem beginning with the original columns x of the form*

$$\begin{pmatrix} c_j^T \\ A_{\cdot j} \\ 0 \end{pmatrix}, \quad \forall j \in \mathcal{C}_n.$$

2. *Corresponding to the left arc from 2 to 1, include columns t and y in the left subproblem of the form*

$$\begin{pmatrix} 1 \\ 0 \\ \tilde{\gamma} \end{pmatrix} \text{ and } \begin{pmatrix} 0 \\ -I \\ \tilde{\Pi} \end{pmatrix},$$

where $\tilde{\gamma}$ and $\tilde{\Pi}$ are information transported by the left arc. The constraints indexed by λ are $\geq$ ones.

3. *Corresponding to the right arc from 2 to 1, include columns in the right subproblem of the form*

$$\begin{pmatrix} -\tilde{l} \\ \tilde{d} \\ \tilde{y} \end{pmatrix},$$

where $(\tilde{l}, \tilde{d}, \tilde{y})$ is information transported by the right arc. In addition, place the term $\tilde{d}w$ in the right-hand side.

4. *Place b in the right-hand side of the left subproblem.*

5. *Make the right subproblem's constraints indexed by π into $\geq$ ones.*

Proof: This theorem is derived from the reverse transform in the same manner that Benders decomposition was derived from D-W decomposition. ∎

Using the forward and reverse transform on the D-W and Benders subproblems and networks, and the duality of those subproblems, we can obtain a duality theorem for the networks. The following definition will help with the mechanics.

Definition 1.21 (Transpose Arcs) *a) up transpose is type left, b) down transpose is type right, c) left transpose is type up, d) right transpose is type down.*

Property 1.22 (Arc Duality) *The transpose of the transpose of an arc type is the same type.*

At this point we introduce our first duality theorem for communication networks.

Theorem 1.23 (Network Duality) *To take the dual of a network g with nodes $\mathcal{N}$ and arcs $\mathcal{A}$, interchange the row and column index sets of each node and transpose all arc types. We call the result g^D and note that the dual of g^D is g. The problem data becomes $(-A, -b, -c)$ so that* minimize *switches with* maximize, *and $\geq$ switches with $\leq$.*

Proof: We can work either way through the sequence

$$g_D \overset{\text{Xform}}{\leftrightarrow} \mathcal{O}(1.2) \overset{\text{Dual}}{\leftrightarrow} \mathcal{O}(1.6) \overset{\text{Xform}}{\leftrightarrow} g_B,$$

which is necessary and sufficient for the short form

$$g_D \overset{\text{Dual}}{\leftrightarrow} g_B.$$

∎

The application of the D-W operator to dual networks as in

$$g_B^D = g_1^D \boxminus [x_1, x_2]$$

creates a new operator for our symbolic calculus.

Definition 1.24 (Benders Operator, $\boxminus$) *The Benders operator maps networks in $\mathcal{G}^1$ into those in $\mathcal{G}^2$ using a partition $[P_1, P_2]$ of C_0, a subproblem index set for node n.*

In words this means:

> *Apply Benders decomposition to the linear program corresponding to the node to be split in the communication network. Partition the variables so that x_1 is in the left (Benders Master) subproblem and x_2 is in the right (Benders Slave) subproblem. The governing algorithm described in Corollary 1.18 is implicitly associated with the resultant communication network.*

This case also begins with one node and no arcs, $g_1 = (\{0\}, \pi, x_1 \cup x_2, \emptyset)$, and

$$g_B = g_1 \boxplus [x_1, x_2],$$

where $g_B = (\{1,2\}, \pi, \pi, x_1, x_2, \{(12),(21)\}, \text{right}, \text{left})$.

Property 1.25 (Benders Inverse Operator) *The inverse operator $\square$ is applicable to the Benders operator as well as the D-W operator:*

$$g_1 = g_1 \boxplus [P_1, P_2] \square \{1,2\},$$

for all partitions $[P_1, P_2]$ of C_0.

Proof: Since we know that $g_1 = (g_1 \boxminus [P_1, P_2]) \square \{1,2\}$ already, by using network duality, it must also hold that $g_1 = g_1^D = (g_1^D \boxminus [P_1, P_2]) \square \{1,2\}$. In addition, since both $g_B^D = g_1^D \boxminus [x_1, x_2]$ and $g_B = g_1 \boxplus [x_1, x_2]$, we now have $g_1 = (g_1 \boxplus [x_1, x_2])^D \square \{1,2\}$. But inversion is not concerned with indices or arc types, so finally, $g_1 = (g_1 \boxplus [x_1, x_2]) \square \{1,2\}$. ∎

1.5 Subproblem Interfaces

The positions of nonzeros in, and the partitioning of, the constraint matrix affects the quantity of information communicated between subproblems. Vacuous columns in a row partition let the corresponding variables be free of constraints. When this

occurs, there is no need to pass information indicating that a variable is free. It is instead possible to make this fact implicit in the formulation of the other subproblem.

The subproblem interface theorem characterizes the portions of subproblem solutions that are exchanged. It is based on the following LP formulation, which uses a partitioning of the columns of (1.2) so that $A_1 = (A_{11} \quad A_{12})$, $A_2 = (A_{21} \quad A_{22})$, and $c^T = (c_1^T \quad c_2^T)$. Thus, the LP of interest is

$$\begin{array}{lll} \min\limits_{\substack{x_1 \geq 0 \\ x_2 \geq 0}} & c_1^T x_1 + c_2^T x_2 = z & \\ \text{s.t.} \quad \pi_1 : & A_{11}x_1 + A_{12}x_2 \geq b_1 & (1.9) \\ \pi_2 : & A_{21}x_1 + A_{22}x_2 \geq b_2, & \end{array}$$

with accompanying starting network g_1 defined as follows:

$$g_1 = (\{0\}, \pi_1 \cup \pi_2, x_1 \cup x_2, \emptyset).$$

Theorem 1.26 (Subproblem Interfaces) *Let us assume that (1.9) is decomposed by the D-W method using the row partition $[\pi_1, \pi_2]$. If $A_{12} = 0$, the dual solution to ψ in the top subproblem is a constant equal to $-c$. The subproblems are formulated as (1.10)′ and (1.11)′ below. Similarly, if $A_{22} = 0$, the primal feasible region for x_{22} is the positive orthant in the bottom subproblem and can be expressed instead using subproblems formulated as (1.10)″ and (1.11)″.*

Proof: With D-W decomposition of (1.9) using the partition $[\pi_1, \pi_2]$, we get the subproblem formulations:

$$\begin{array}{lll} \min\limits_{\substack{l \geq 0 \\ x_{11}, x_{12}}} & c_1^T x_{11} + c_2^T x_{12} = z_1 & \\ \text{s.t.} \quad \theta : & \tilde{g}^T l = 1 & \\ \psi_1 : & \tilde{X}_{21} l - I x_{11} = 0 & (1.10) \\ \psi_2 : & \tilde{X}_{22} l - I x_{12} = 0 & \\ \pi_1 : & A_{11}x_{11} + A_{12}x_{12} \geq b_1, & \end{array}$$

and

$$\begin{array}{lll} \min\limits_{\substack{x_{21} \geq 0 \\ x_{22} \geq 0, w}} & \tilde{\delta} w = z_2. & \\ \text{s.t.} \quad \upsilon : & \tilde{\psi}_1 x_{21} + \tilde{\psi}_2 x_{22} + \tilde{\delta} w \geq -\tilde{\theta} & (1.11) \\ \pi_2 : & A_{21}x_{21} + A_{22}x_{22} \geq b_2. & \end{array}$$

Case 1: If $A_{12} = 0$ the solution for ψ_2 always equals $-c_2$ by the dual constraints. Substituting $\tilde{X}_{22}l$ for x_{12} in the objective row of (1.10) and fixing $\tilde{\psi}_2 = -c_2$ in (1.11), we get

$$\begin{array}{lll} \min\limits_{l\geq 0, x_{11}} & c_2^T\tilde{X}_{22}l + c_1^T x_{11} = z_1 & \\ \text{s.t.} \quad \theta: & \tilde{g}^T l = 1 & \\ \psi_1: & \tilde{X}_{21}l - Ix_{11} = 0 & \\ \pi_1: & A_{11}x_{11} \geq b_1, & \end{array} \qquad (1.10)'$$

and

$$\begin{array}{lll} \min\limits_{\substack{x_{21}\geq 0 \\ x_{22}, w}} & c_2^T x_{22} + \tilde{\delta}w = z_2. & \\ \text{s.t.} \quad \upsilon: & \tilde{\psi}_1 x_{21} + \tilde{\delta}w \geq -\tilde{\theta} & \\ \pi_2: & A_{21}x_{21} + A_{22}x_{22} \geq b_2. & \end{array} \qquad (1.11)'$$

Case 2: If $A_{22} = 0$ the feasible region for x_{22} is the positive orthant. Therefore, by moving the non-negativity constraints for these columns to the top subproblem and eliminating those columns from the bottom, we get

$$\begin{array}{lll} \min\limits_{\substack{l\geq 0 \\ x_{11}, x_{12}\geq 0}} & c_1^T x_{11} + c_2^T x_{12} = z_1 & \\ \text{s.t.} \quad \theta: & \tilde{g}^T l = 1 & \\ \psi_1: & \tilde{X}_{21}l - Ix_{11} = 0 & \\ \pi_1: & A_{11}x_{11} + A_{12}x_{12} \geq b_1, & \end{array} \qquad (1.10)''$$

and

$$\begin{array}{lll} \min\limits_{x_{21}\geq 0, w} & \tilde{\delta}w = z_2. & \\ \text{s.t.} \quad \upsilon: & \tilde{\psi}_1 x_{21} + \tilde{\delta}w \geq -\tilde{\theta} & \\ \pi_2: & A_{21}x_{21} \geq b_2. & \end{array} \qquad (1.11)''$$

In summary,

If $A_{12} = 0$ then (1.10)(1.11) $\equiv$ (1.10)′(1.11)′, and

if $A_{22} = 0$ then (1.10)(1.11) $\equiv$ (1.10)″(1.11)″.

■

We have also proved that the communication graph is not affected by alternative subproblem formulations, hence alternative subproblem interfaces: $g_1 \boxminus [\pi_1, \pi_2] = g_2$ where

$$g_2 = (\{1,2\}, \pi_1, \pi_2, x_1 \cup x_2, x_1 \cup x_2, \{(12),(21)\}, \text{down, up}).$$

The dual of the Subproblem Interface Theorem yields the following corollary. The subproblem formulations are left as an exercise.

Corollary 1.27 (Subproblem Interfaces) *Let us assume that (1.9) is decomposed by the Benders method using the column partition $[x_1, x_2]$. If $A_{21} = 0$, the primal solution to y_2 in the left subproblem is a constant equal to $-b_2$. Similarly, if $A_{22} = 0$, the dual feasible region for π_{22} in the right subproblem is the positive orthant.*

In the sequel, unnecessary variables and constraints will be dropped when submatrices are equal to zero.

1.6 Summary

We have reviewed the theory of Dantzig-Wolfe and Benders decomposition, and found their subproblem formulations and their algorithms to be duals of each other. In the process, we introduced the symbolic space of *communication networks* with one and two nodes, $\mathcal{G}^1$ and $\mathcal{G}^2$ respectively. The algebraic decomposition of LP subproblems is equated to the splitting of node index sets in communication networks. The set $\mathcal{G}^1$ contains one network which is self dual, and the set $\mathcal{G}^2$ contains two networks which are duals of each other. In the next chapter we will explore the span of decomposition schemes possible under our defined operators. It is $\mathcal{G}^N$. Since higher dimensional schemes are constructed upon lower dimensional ones, and since the two entries of $\mathcal{G}^2$ are duals, this automatically divides all of $\mathcal{G}^N$ in half. Every scheme in one half has a dual scheme in the other; except for $\mathcal{G}^1$, which lies in both (or neither).

We have shown that the duality of linear programming translates directly to a duality for networks. The next chapter characterizes the space of all subproblem

formulations and likewise their accompanying communication networks. Once characterized, we can give the transforms and the duality of all networks.

Finally, the pattern of zeros and nonzeros in the constraint matrix affects the interfaces between subproblems, and even their formulations. By investigating these patterns, we can drastically reduce the quantity of information communicated over the network for sparse problems.

Chapter 2

Characterizing Communication Networks

EFFICIENT use of a parallel computer requires that there be many subproblems that can be solved independently. Having completed the derivations of the top, bottom, left and right subproblems, and the communication diagrams describing their interactions, we now embark on an exploration of the versatility of the D-W and Benders partitioning operators and their use in creating many subproblems for a parallel computer to solve. The following sections describe a variety of decomposition schemes that can be generated with partially ordered sets of partitions within partitions. Each scheme alone is capable of performing the oracle on the original problem, $\mathcal{O}(1.1)$.

This chapter concentrates solely on partitions, subproblems, and networks; a skeletal framework and on which to attach the muscles, the algorithms. The next chapter covers the parallel decomposition oracle, which is a relaxed form of nesting oracles. Naturally, we get a serial oracle from the parallel one when using only one processor. For now we take faith in nesting the oracle and proceed.

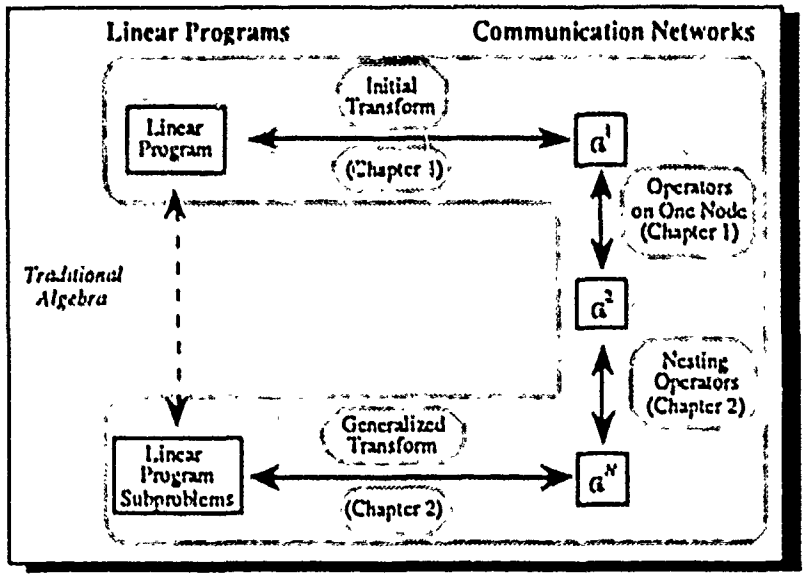

Figure 2.1: Symbolic Decomposition covered by Chapter Two.

2.1 Nested Decomposition

In the previous chapter we described decomposition in terms of operations on communication networks that form new, higher-ordered networks. The term *nested*, in the title of this section, refers to the practice of embedding one thing within another. Using our operators, we can nest partitions and oracles with a sequence of slices. For the present, we will nest only D-W decomposition and state simply that the dual of each operation we perform applies equally well in the context of Benders decomposition. As an extension to traditional decomposition, we introduce cross-nesting, the practice of using both D-W and Benders decomposition on the same problem.

We consider three variations of nesting the D-W and Benders operators in the network g_D, which together with their dual versions comprise the complete set of communication networks on three nodes: $\mathcal{G}^3$. The three variations are: splitting the top with $\boxminus$ and splitting the bottom with $\boxminus$ and $\boxplus$. We demonstrate the algebraic derivations of the subproblems and note that we can successfully use the defined operators to chronicle the mapping of g_D into $\mathcal{G}^3$. As a summary, we present the

general forward transformation, and apply our decomposition operators to networks in $\mathcal{G}^N$.

Early references on nested decomposition are [Dan73], [Gla73], and [Ho74]. Later, Abrahamson [AbrS3] and Wittrock [WitS3] enhanced the dual version or Nested-Benders Decomposition. Here, we take a very generic view of nesting, considering more fashions of subproblems.

Before starting, let us redefine (1.1) with $A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}$ and $b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$ so that we now wish to find an oracle for solving

$$\begin{array}{lll} \min\limits_{x \geq 0} & & c^T x = z \\ \text{s.t.} & \pi_1: & A_1 x \geq b_1 \\ & \pi_2: & A_2 x \geq b_2 \\ & \pi_3: & A_3 x \geq b_3, \end{array} \tag{2.1}$$

with its corresponding network having one node and no arcs,

$$g_1 = (\{0\}, \pi_1 \cup \pi_2 \cup \pi_3, x, \emptyset).$$

The last chapter covered the two cases of applying $\boxminus$ and $\boxplus$ to g_1 to generate g_D and g_B. In turn we will new apply the same operators to g_D and g_B. The main difference between splitting the node in g_1 and one in either g_D or g_B is that the two latter types have incident arcs. What do we do with these incident arcs? In the next two lemmas, we adopt the convention that:

> *When a node in g_D is split using $\boxminus$, those arcs once incident to the split node will be made incident to the new top node.*

This convention makes communication networks have tree structures. Splitting a bottom node extends a branch of the network, while splitting the top node starts a new branch.

Lemma 2.1 ($\boxminus$ on the Bottom Node) *The D-W operator can be applied to the bottom node using the expression*

$$g_3 = g_1 \boxminus [\pi_1, [\pi_2, \pi_3]],$$

where

$$g_3 = (\{1,3,4\}, \pi_1, \pi_2, \pi_3, x, x, x, \{(13),(34),(31),(43)\}, \text{down, down, up, up}).$$

Proof: Let $[\pi_1, [\pi_2, \pi_3]]$ be a partition of the original row index set $\mathcal{R}$. Decompose the LP formulated in (2.1) into three subproblems using two applications of the D-W operator. The resulting subproblems will exhibit a linear communication structure as in Figure 2.2. The dotted supernode 2 represents an *implicit subproblem* that has itself been decomposed into nodes 3 and 4

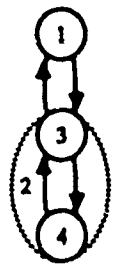

Figure 2.2: Spitting the bottom node.

The first application of the operator groups the bottom two indices together in the second partition:

$$g_1 \boxminus [\pi_1, \pi_2 \cup \pi_3] = g_2,$$

where the resulting subproblems are

$$\begin{array}{lll} \min\limits_{\substack{x_1 \\ l_1 \geq 0}} & & c_1^T x_1 = z_1 \\ \text{s.t.} \quad \theta_1 : & \tilde{g}_2^T l_1 & = 1 \\ \psi_1 : & \tilde{X}_2 l_1 - I x_1 & = 0 \\ \pi_1 : & A_1 x_1 & \geq b_1, \end{array} \tag{2.2}$$

and

$$\begin{array}{lll} \min\limits_{x_2 \geq 0, w_2} & & \tilde{\delta}_1 w_2 = z_2 \\ \text{s.t.} \quad \upsilon_2 : & \tilde{\psi}_1^T x_2 + \tilde{\delta}_1 w_2 & \geq -\tilde{\theta}_1 \\ \pi_2 : & A_2 x_2 & \geq b_2 \\ \pi_3 : & A_3 x_2 & \geq b_3, \end{array} \tag{2.3}$$

and $g_2 = (\{1,2\}, \pi_1, \pi_2 \cup \pi_3, x, x, \{(12),(21)\}, \text{down, up})$. The second application of the D-W operator uses the partition $[\pi_2, \pi_3]$ to slice horizontally through (2.3). The resulting subproblem formulations are

$$\begin{array}{lll} \min\limits_{\substack{l_2 \geq 0 \\ x_2, w_2}} & & \bar{\delta}_1 w_2 = z_2 \\ \text{s.t.} \quad v_2: & \bar{\psi}_1^T x_2 + \bar{\delta}_1 w_2 & \geq -\bar{\theta}_1 \\ \theta_2: & \bar{g}_3^T l_2 & = 1 \\ \psi_2: & \bar{X}_3 l_2 - I x_2 & = 0 \\ \pi_2: & A_2 x_2 & \geq b_2, \end{array} \tag{2.4}$$

and

$$\begin{array}{lll} \min\limits_{x_3 \geq 0, w_3} & & \bar{\delta}_2 w_3 = z_3 \\ \text{s.t.} \quad v_3: & \bar{\psi}_2^T x_3 + \bar{\delta}_2 w_3 & \geq -\bar{\theta}_2 \\ \pi_3: & A_3 x_3 & \geq b_3, \end{array} \tag{2.5}$$

with the v_2 constraints included in the top partition.

The final network that corresponds to the subproblem triple (2.2)(2.4)(2.5) is g_3 and can be compared to that pictured in Figure 2.2. ∎

By reordering the nesting we just used, we induce a different communication pattern from the one above. When we split the top node with $\boxminus$, we spawn a new branch in the network.

Lemma 2.2 ($\boxminus$ on the Top Node) *The D-W operator can be applied to the top node using the expression*

$$g_1 \boxminus [[\pi_1, \pi_2], \pi_3] = g_3,$$

where

$$g_3 = (\{3,4,1\}, \pi_1, \pi_2, \pi_3, x, x, x, \{(34),(32),(43),(23)\}, \text{down, down, up, up}).$$

Proof: Apply the partition $[[\pi_1, \pi_2], \pi_3]$ to (2.1) to get the system of subproblems (2.6)(2.7)(2.5), where

$$\begin{array}{lll} \min\limits_{\substack{l_1 \geq 0 \\ l_2 \leq 0 \\ x_1}} & & c^T x_1 = z_1 \\ \text{s.t.} & \theta_1 : & \bar{g}_2^T l_1 = 1 \\ & \theta_2 : & \bar{g}_3^T l_2 = 1 \\ & \psi_1 : & \bar{X}_2 l_1 - I x_1 = 0 \\ & \psi_2 : & \bar{X}_3 l_2 - I x_1 = 0 \\ & \pi_1 : & A_1 x_1 \geq b_1, \end{array} \tag{2.6}$$

and

$$\begin{array}{lll} \min\limits_{x_2, w_2} & & \bar{\delta}_1 w_2 = z_2 \\ \text{s.t.} & \upsilon_2 : & \bar{\psi}_1^T x_2 + \bar{\delta}_1 w_2 \geq -\bar{\theta}_1 \\ & \pi_2 : & A_2 x_2 \geq b_2, \end{array} \tag{2.7}$$

with the accompanying communication network g_3 as shown in Figure 2.3. ∎

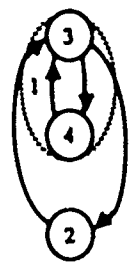

Figure 2.3: Splitting the top node.

The term *cross splitting* will be used to describe the process of nesting Benders decomposition within D-W decomposition, and vice versa. We use the following definition to identify such a condition.

Definition 2.3 (Cross Splitting) *A node is cross split when the* ⊟ *operator is applied and it has an incoming right arc, and similarly, when the* ⊞ *operator is applied and it has an incoming down arc.*

We will not cross split added constraints and variables that function as partial representations of the feasible regions of still other subproblems (incoming up and

left arcs). This possibility would take us beyond the scope of this thesis. However, the extra subproblem data that implement objective or right-hand side modifications are considered valid places for cross splitting. For instance, it is valid to partition the columns of the D-W bottom subproblem (1.5) but not the top one (1.4). Figure 2.4 illustrates the communication network that results from cross splitting the bottom node in g_D.

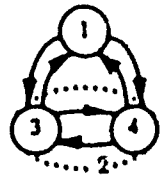

Figure 2.4: Cross splitting the bottom node.

First, we repeat the LP fomulation used for the Subproblem Interface Theorem:

$$\begin{array}{lll} \min\limits_{\substack{x_1 \geq 0 \\ x_2 \geq 0}} & & c_1^T x_1 + \; c_2^T x_2 = z \\ \text{s.t.} & \pi_1 : & A_{11}x_1 + A_{12}x_2 \geq b_1 \\ & \pi_2 : & A_{21}x_1 + A_{22}x_2 \geq b_2, \end{array} \tag{2.8}$$

and its accompanying network is

$$g_1 = (\{0\}\pi_1 \cup \pi_2, x_1 \cup x_2, \emptyset).$$

Figure 2.4 completes the depiction of all networks in $\mathcal{G}^3$ that can be generated from g_D. The changes made to g_D to get this network do not follow our previous convention of making arcs once incident to the split node incident now to the new left node. Because we have cross split, the arcs in question must be duplicated and made incident to *both* new nodes. This is evident from the subproblem formulations and an application of the forward transform.

Lemma 2.4 ($\boxplus$ on the Bottom Node) *The Benders operator $\boxplus$ can be applied to the bottom node using the expression*

$$(g_1 \boxminus [\pi_1, \pi_2]) \boxplus [x_1, x_2] = g_3,$$

where

$$g_3 = (\{1,3,4\}, \pi_1, \pi_2, \pi_2, x_1 \cup x_2, x_1, x_2,$$

$$\{(13),(14),(31),(41),(23),(32)\}, \text{down, down, up, up, right, left}).$$

Proof: First we create modified versions of the top and bottom subproblems on which to demonstrate. To do this, apply D-W decomposition to (2.8) as the following expression suggests:

$$g_2 = g_1 \boxminus [\pi_1, \pi_2],$$

where $g_2 = (\{1,2\}, \pi_1, \pi_2, x_1 \cup x_2, x_1 \cup x_2, \{(12),(21)\}, \text{down, up})$. The top and bottom subproblem formulations are:

$$\begin{array}{lll} \min\limits_{\substack{l \geq 0 \\ x_{11}, x_{12}}} & & c_1^T x_{11} + c_2^T x_{12} = z_0 \\ \text{s.t.} \quad \theta: & \tilde{g}_2^T l & = 1 \\ \psi_1: & \tilde{X}_{21} l - I x_{11} & = 0 \\ \psi_2: & \tilde{X}_{22} l - I x_{12} & = 0, \\ \pi_1: & A_{11} x_{11} + A_{12} x_{12} & \geq b_1, \end{array} \tag{2.9}$$

and

$$\begin{array}{lll} \min\limits_{\substack{w_{21}, w_{22} \\ x_{21} \geq 0, x_{22} \geq 0}} & & \tilde{\delta} w_1 + \tilde{\delta} w_2 = \tilde{z}_2 \\ \text{s.t.} \quad \upsilon_1: & \tilde{\psi}_1^T x_{21} + \tilde{\delta} w_1 & \geq -\tilde{\theta} \\ \upsilon_2: & \tilde{\psi}_2^T x_{22} + \tilde{\delta} w_2 & \geq 0 \\ \pi_2: & A_{21} x_{21} + A_{22} x_{22} & \geq b_2. \end{array} \tag{2.10}$$

Note that some liberty was taken in the formulation of (2.10) by implementing the objective modification with two added variables and constraints instead of one of each.

Continue by crossing Benders decomposition on (2.10) with the partition $[x_{21}, x_{22}]$. The left and right subproblems are

$$\begin{array}{lll} \min\limits_{\substack{x_{21}\geq 0, w_{21} \\ y_2, t_2}} & \bar{\delta} w_1 + t = z_{21} & \\ \text{s.t.} \quad v_1: & \bar{\psi}_1^T x_{21} + \bar{\delta} w_1 & \geq -\bar{\theta}_2 \\ \pi_{21}: & A_{21} x_{21} - I y & = b_2 \\ \lambda_1: & \bar{\Pi}_{22}^T y + \bar{\gamma}_{23} t & \geq 0, \end{array} \tag{2.11}$$

and

$$\begin{array}{lll} \min\limits_{u_{22}, x_{22}\geq 0, w_{22}} & -\bar{t} u_{22} & \bar{\delta} w_2 = z_{22}, \\ \text{s.t.} \quad v_{22}: & \bar{\psi}_2^T x_{22} + \bar{\delta} w_2 & \geq 0 \\ \pi_{22}: & \bar{y} u_{22} + A_{22} x_{22} & \geq 0 \\ \omega_{22}: & \bar{d} u_{22} & = \bar{d}, \end{array} \tag{2.12}$$

where w_1 has followed x_{21}, and w_2 has followed x_{22}. ∎

To summarize this cross splitting example, we began with the linear program (2.8), applied Dantzig-Wolfe decomposition to obtain the subproblem system (2.9)(2.10), then applied Benders to the bottom subproblem. This was made possible by expressing the added structure for objective modification with multiple constraints, instead of a single one. The final communication network is g_3 and is shown in Figure 2.4. The final subproblem system is (2.9)(2.11)(2.12).

The following corollary formally notes that the other networks in $\mathcal{G}^3$ are duals of the three above.

Corollary 2.5 (Nested Duality) *Theorems 2.2, 2.1 and 2.4 apply also to splitting the nodes of g_B with the words top and bottom replaced by left and right, and switching $\boxminus$ with $\boxplus$ and row partitions with column partitions.*

Proof: This statement follows from the network duality theorem. ∎

In conclusion, the operations on g_D and g_B have demonstrated how to split top, bottom, left and right nodes. The distinguishing feature of these nodes was that they had incident arcs from either above or below, but not both. We look forward to having our operators applied to any node in a network.

2.2 Characterizing $\mathcal{G}^3$

As we have stated earlier, the previous three operations on the network g_D are defined to be the only valid ones. Table 2.1 enumerates the six mappings from $\mathcal{G}^2$ to $\mathcal{G}^3$, and Figure 2.5 has them drawn out.

The following are the descriptions for column headings in Table 2.1. The table represents the results of a boolean function on the column headings. Each heading can take one of two values.

First Split Uses: Either the ⊟ or the ⊞ operator is applied to g_1 to get either g_D or g_B.

Second Split Uses: The operator used for the second slice.

Second Splits Node: The node split on the second slice. We use number 1 to indicate the top or left node, and the number 2 to indicate the bottom or right node.

	First Split Uses	Second Split Uses	Second Splits Node	
1	⊟	⊟	1	Valid
2	⊟	⊟	2	Valid
	⊟	⊞	1	Invalid
3	⊟	⊞	2	Valid
	⊞	⊟	1	Invalid
4	⊞	⊟	2	Valid
5	⊞	⊞	1	Valid
6	⊞	⊞	2	Valid

Table 2.1: The Elements of $\mathcal{G}^3$.

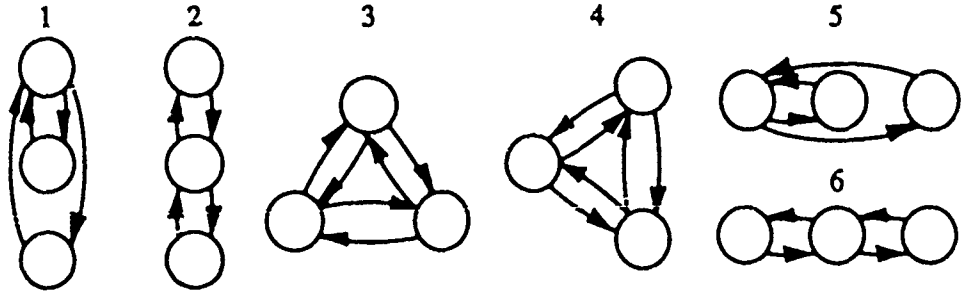

Figure 2.5: The elements of $\mathcal{G}^3$.

2.3 Characterizing $\mathcal{G}^N$

By characterizing all decomposition schemes that are attainable with our defined set of operators, we will be able to state a parallel decomposition oracle that holds over the entire space. First we need to handle one more case for splitting nodes, then we can show that our operators are well defined for any node by demonstrating their validity on the most general case. We need to define a "middle" node and how to split it.

Definition 2.6 (Middle Node) *A node $n \in \mathcal{N}$ is a middle node if it has incoming and outgoing up or left arcs.*

Our convention on incident arcs remaining incident to the new top node means that splitting a middle node adds a new branch to the network.

Lemma 2.7 ($\boxminus$ on a Middle Node) *The D-W operator can be applied to a middle node with no incoming left arcs.*

Proof: We start by redefining the LP (2.4) by the partition $A_2 = \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix}$ and $b_2 = \begin{pmatrix} b_{12} \\ b_{22} \end{pmatrix}$ to arrive at

$$\begin{array}{lll} \min\limits_{\substack{l_2 \geq 0 \\ x_2, w_2}} & & \tilde{\delta}_1 w_2 = z_2 \\ \text{s.t.} \quad v_2: & \tilde{\psi}_1^T x_2 + \tilde{\delta}_1 w_2 \geq -\tilde{\theta}_1 \\ \theta_2: & \tilde{g}_3^T l_2 = 1 \\ \psi_2: & \tilde{X}_3 l_2 - I x_2 = 0 \\ \pi_{21}: & A_{21} x_2 \geq b_{21} \\ \pi_{22}: & A_{22} x_2 \geq b_{22}. \end{array} \tag{2.13}$$

Next, the expression

$$g_1 \boxminus [[\pi_1, [\pi_{21}, \pi_{22}]], \pi_3]$$

suggests that $\mathcal{O}(2.13)$ can in turn be performed by the D-W Method using $\mathcal{O}(2.14)$ and $\mathcal{O}(2.15)$, where

$$\begin{array}{lll} \min\limits_{\substack{l_2 \geq 0 \\ x_2, w_2}} & & \tilde{\delta}_1 w_2 = z_2 \\ \text{s.t.} \quad v_2: & & \tilde{\psi}_1^T x_2 + \tilde{\delta}_1 w_2 \geq -\tilde{\theta}_1 \\ \theta_2: & \tilde{g}_3^T l_{21} & = 1 \\ \psi_2: & \tilde{X}_3 l_{21} - I x_2 & = 0 \\ \theta_{22}: & \tilde{g}^T{}_{22} l_{22} & = 1 \\ \psi_{22}: & \tilde{X}_{22} l_{22} - I x_2 & = 0 \\ \pi_{21}: & A_{21} x_2 & \geq b_{21} \end{array} \tag{2.14}$$

and

$$\begin{array}{lll} \min\limits_{x_{22}, w_{22}} & & \tilde{\delta}_{22} w_{22} = z_{22} \\ \text{s.t.} \quad v_{22}: & \tilde{\psi}^T{}_{22} x_{22} + \tilde{\delta}_{22} w_{22} \geq -\tilde{\theta}_{22} \\ \pi_{22}: & A_{22} x_2 \geq b_{22}. \end{array} \tag{2.15}$$

The forward transform gives us the communication network in Figure 2.6. ∎

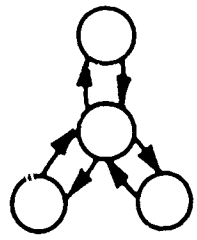

Figure 2.6: Splitting a middle node.

In conclusion, our networks get longer from the bottom and bushier from the top and the middle.

By proving a partial ordering property, we can use terms like *upper, above, lower, and below* as relations between nodes. The > symbol is used to indicate an ordering between two nodes.

Property 2.8 (Partial Node Ordering) *There is a partial ordering on the nodes of networks in $\mathcal{G}^N$, following the rule that $n_1 > n_2$ if there is an up arc from n_2 to n_1. Node n_1 is said to be above n_2.*

Proof: There is a directed graph of up and left arcs that spans the nodes of $g \in \mathcal{G}^N$ and it has no directed circuits. Hence, it induces a partial order on up arcs. ∎

We now summarize the forward transform from Chapter 1 as a seven-step process. The first step defines a node for each subproblem, and the other steps define the other four tuples in a communication network according the subproblem indices: $\bar{\mathcal{R}}_n$ and $\bar{\mathcal{C}}_n$.

Theorem 2.9 (Generalized Transform) *The transform from a system of N subproblems to a communication network with N nodes is a seven step process:*

1. *For each subproblem, define a* node *in the set $\mathcal{N}$.*
2. *For each node $n \in \mathcal{N}$, define the elements of the* row index set $\mathcal{R}_n = \bar{\mathcal{R}}_n \cap \mathcal{R}$.
3. *For each node $n \in \mathcal{N}$, define the elements of the* column index set $\mathcal{C}_n = \bar{\mathcal{C}}_n \cap \mathcal{C}$.

4. *For each subproblem* $n_1 \in \mathcal{N}$ *containing* constraints indexed by θ and ψ, *and for every other subproblem* $n_2 \in \mathcal{N}$, *that provides information for those constraints, define an arc* $(n_2\, n_1) \in \mathcal{A}$ *of type* $\mathcal{T}_{n_2 n_1} = up$.

5. *For each subproblem* $n_1 \in \mathcal{N}$ *containing* constraints indexed by v, *and for every other subproblem* $n_2 \in \mathcal{N}$ *that provides information for those constraints, define an arc* $(n_2\, n_1) \in \mathcal{A}$ *of type* $\mathcal{T}_{n_2 n_1} = down$.

6. *For each subproblem* $n_1 \in \mathcal{N}$ *containing* variables named y and t, *and for every other subproblem* $n_2 \in \mathcal{N}$ *that provides information for those columns, define an arc* $(n_2\, n_1) \in \mathcal{A}$ *of type* $\mathcal{T}_{n_2 n_1} = left$.

7. *For each subproblem* $n_1 \in \mathcal{N}$ *containing* variables named u, *and for every other subproblem* $n_2 \in \mathcal{N}$ *that provides information for those columns, define an arc* $(n_2\, n_1) \in \mathcal{A}$ *of type* $\mathcal{T}_{n_2 n_1} = right$.

Proof: First, the theorem holds for schemes involving only D-W decomposition, since we know that the transform is correct for a top, bottom, or middle node, as already demonstrated. Second, by network duality, the theorem holds for schemes involving only Benders decomposition. Finally, when both types of decomposition are represented in the same network, we can transform a node with adjacent horizontal *and* vertical arcs because the arcs have independent effects on the formulation. The added variables and constraints of a subproblem interact only through their incidence to the original primal and dual variables, x and π. ∎

Definition 2.10 (Inverse Operator on $\mathcal{G}^N$) *The inverse operation on* $\mathcal{G}^N$ *is defined as one being reversible by a series of applications of the D-W and/or Benders operators.*

$$\mathcal{G}^N \square \mathcal{N}^* \rightarrow \mathcal{G}^{N-|\mathcal{N}^*|+1},$$

where $|\mathcal{N}^*| \leq N$. *For the specific networks* $g_2 \in \mathcal{G}^N$ *and* $g_1 \in \mathcal{G}^{N-|\mathcal{N}^*|+1}$, *to take an inverse using any* $\mathcal{N}^* \subseteq \mathcal{N}_2$, *collapse all the nodes into one, and redefine all of the incident arcs as follows:*

1. $\mathcal{N}_1 \leftarrow \mathcal{N}_2 - \mathcal{N}^\bullet + \{n\}, \quad n \notin \mathcal{N}_1,$
2. $\mathcal{R}_n = \bigcup_{n_1 \in \mathcal{N}^\bullet} \mathcal{R}_{n_1},$
3. $C_n = \bigcup_{n_1 \in \mathcal{N}^\bullet} C_{n_1},$
4. $\mathcal{A}_1 \leftarrow \mathcal{A}_2 - \bigcup_{n_1 \in \mathcal{N}^\bullet} \{(n_1\ n_2)\} - \bigcup_{n_2 \in \mathcal{N}^\bullet} \{(n_1\ n_2)\}$
   $+ \bigcup_{\substack{(n_1\ n_2) \in \mathcal{A}_2 \\ n_1 \in \mathcal{N}^\bullet}} \{(n\ n_2)\} + \bigcup_{\substack{(n_1\ n_2) \in \mathcal{A}_2 \\ n_2 \in \mathcal{N}^\bullet}} \{(n_1\ n)\},$
5. $\mathcal{T}_{n_1\ n} = \mathcal{T}_{n_1\ n_2}, \forall (n_1\ n) \in \mathcal{A}_1$ *and* $(n_1\ n_2) \in \mathcal{A}_2,$

where we have used + and − to mean set union and subtraction. Note that we will not allow sets to contain duplicate elements.

The inverse operation is defined in terms of being reversible. We now give two conditions on the set of nodes to collapse $\mathcal{N}^\bullet$, that offer this feature. First, define the following terms:

up connected nodes: For some graph $g \in \mathcal{G}^N$, the nodes in the subset $\mathcal{N}^\bullet \in \mathcal{N}$ are said to be up connected if and only if for all n_1, n_2 in $\mathcal{N}^\bullet$ there exists an n_1, n_2 undirected path on up arcs that visits only nodes in $\mathcal{N}^\bullet$.

left connected nodes: The dual of a network on up connected nodes.

Lemma 2.11 ($\square$ for Connected Nodes) *If for some network $g \in \mathcal{G}^N$, the nodes in $\mathcal{N}^\bullet \in \mathcal{N}$ are either up or left connected, then the effects of the inverse operation*

$$g' = g \square \mathcal{N}^\bullet$$

can be reversed by a series of D-W or Benders operations, respectively.

Proof: By induction on the number of nodes in $\mathcal{N}^\bullet$.

1. Show it for $|\mathcal{N}^\bullet| = 2$. Take from a network $g \in \mathcal{G}^N$ two nodes n_1 and n_2 which are up connected. We have shown in Chapter One the inverse operation used on the networks in $\mathcal{G}^2$. This step is reversible because when splitting the aggregate node, incident arcs can be replaced to their original positions by choosing the proper partition.

2. Assume that the operator holds for $|\mathcal{N}^\bullet| = N - 1$.

3. Show it for $|\mathcal{N}^\bullet| = N$. Take a set of *up connected* nodes $\mathcal{N}^\bullet$ where $|\mathcal{N}^\bullet| = N$. We have shown that any two up connected nodes in $\mathcal{N}^\bullet$ can be joined into one, thus reducing the order of the network by one node. By the induction assumption, the operator then holds for any $\mathcal{N}^\bullet$.

∎

Lemma 2.12 ($\square$ for Unconnected Nodes) *For some network $g \in \mathcal{G}^N$ and some subset $\mathcal{N}^\bullet = \{n_1, n_2\} \subset \mathcal{N}$, n_1 and n_2 are both up connected to n_3, then the effects of the inverse operation*

$$g' = g \,\square\, \mathcal{N}^\bullet$$

are reversible by a series of splitting operations.

Proof: We use a bidirectional sequence of inverse and splitting operations on networks in $\mathcal{G}^1$ to $\mathcal{G}^3$ to show that the necessary node configurations can be achieved. For larger networks, any arcs not incident to these three nodes are left unaffected, by design. Incident arcs are either between the three nodes, in which case they are covered by the operators on $\mathcal{G}^3$, or they pass outside the three nodes, in which case their sources and destinations within the three nodes can be set by choosing the splitting partitions properly. Transforms from one network to another and back again are shown in Figure 2.7 and explained below.

$\square$: Collapse node 2 into node 1.

$\square$: Collapse node 3 into node 12.

$\boxminus$: Split node 123 with a partition of the rows.

∎

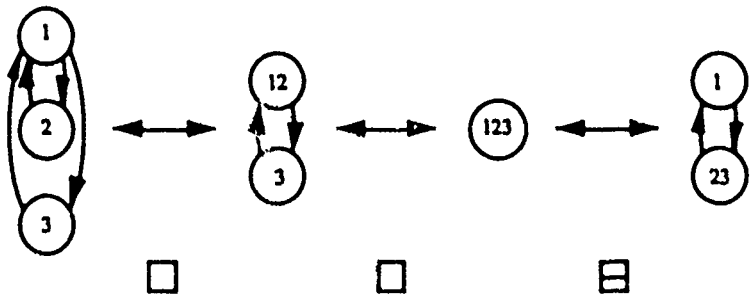

Figure 2.7: Proof of moving up arcs.

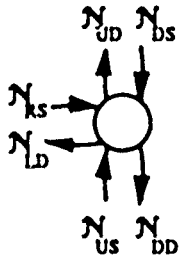

Figure 2.8: The generic node.

Definition 2.13 (Generic Node) *As pictured in Figure 2.8, the generic node has incident arcs such that:*

1. *incoming Up arcs have Sources in $\mathcal{N}_{US}$,*
2. *incoming Down arcs have Sources in $\mathcal{N}_{DS}$, and*
3. *incoming Right arcs have Sources in $\mathcal{N}_{RS}$,*
4. *outgoing Up arcs are Destined for $\mathcal{N}_{UD}$.*
5. *outgoing Down arcs are Destined for nodes in $\mathcal{N}_{DD}$,*
6. *outgoing Left arcs are Destined for nodes in $\mathcal{N}_{LD}$,*

and no others.

Lemma 2.14 (Generic Node of $\mathcal{G}^N$) *Each node and its incident arcs of a network in $\mathcal{G}^N$ can be described within the structure of the generic node or its dual.*

Proof: By induction on the number of nodes.

1. The lemma holds easily for networks in $\mathcal{G}^1$ and $\mathcal{G}^2$.

2. Assume that the lemma holds for all nodes of networks in $\mathcal{G}^N$.

3. By the definitions of the $\boxminus$ and $\boxplus$ operators, no node may be cross split if it has an incoming up or left arc. Only the $\boxplus$ operation may be used on the generic node. Therefore, only left and right arcs may be added when splitting it, which takes the network from $\mathcal{G}^N$ to $\mathcal{G}^{N+1}$. The dual holds for the dual of the generic node.

∎

Lemma 2.15 ($\mathcal{G}^N \boxdot \mathcal{N}^* \rightarrow \mathcal{G}^4$) *Pick a node n in a network $g_N \in \mathcal{G}^N$. Using the $\boxdot$ operator, this network can be reduced to the four-node network in Figure 2.9, its dual, or some special case of either.*

Proof: It is sufficient to prove that the connected node sets of the generic node can be reduced to the three nodes so that

a) $\mathcal{N}_{UD} = \mathcal{N}_{DS}$, $\mathcal{N}_{LD} = \mathcal{N}_{RS}$, $\mathcal{N}_{US} = \mathcal{N}_{DD}$.

b) $\mathcal{N}_{UD} \neq \mathcal{N}_{LD}$, $\mathcal{N}_{LD} \neq \mathcal{N}_{US}$, $\mathcal{N}_{US} \neq \mathcal{N}_{UD}$, and

c) $\mathcal{N}_{UD} = \{n_{UD}\}$, $\mathcal{N}_{LD} = \{n_{LD}\}$, $\mathcal{N}_{US} = \{n_{US}\}$, and

We know a) holds since all communication networks are symmetric; for every arc $(n_1\, n_2)$ there is a corresponding arc $(n_2\, n_1)$. We know b) holds because the nodes are partially ordered. We know c) holds because:

- any nodes on left arcs will collapse to either one node in an up tree, or the two nodes: n_{LD} and n, and
- there is only one tree on up arcs containing n implying that $\mathcal{N}_{UD}$ and $\mathcal{N}_{US}$ can be collapsed to one node each.

∎

We now define the generalized Dantzig-Wolfe operator on the generic node.

Definition 2.16 ($\boxminus$ on the Generic Node) *To apply the $\boxminus$ operation to a generic node $n \in \mathcal{N}_N$, from a network $g_N \in \mathcal{G}^N$ by, $g_{N+1} = g_N \boxminus [P_1, P_2]$, we define the transition of each tuple in g_N to that in g_{N+1}.*

1. *Node n is discarded and two new nodes are added: $\mathcal{N}_{N+1} = \mathcal{N}_N - \{n\} + \{n_1, n_2\}$, where $n_1, n_2 \notin \mathcal{N}_N$.*
2. *All arcs incident to node n are discarded. Of those, the vertical ones are linked to nodes n_1, and the horizontal ones are duplicated and incident to both nodes n_1 and n_2:*

$$\begin{aligned}
\mathcal{A}_{N+1} \;=\; & \mathcal{A}_N - \bigcup_{n' \in \mathcal{N}_N} \{(n'\,n), (n\,n')\} \\
& + \bigcup_{n' \in \mathcal{N}_{US}} \{(n'\,n_1)\}, + \bigcup_{n' \in \mathcal{N}_{DD}} \{(n_1\,n')\}, \;\; \text{if } y, t \in P_1 \\
& + \bigcup_{n' \in \mathcal{N}_{US}} \{(n'\,n_2)\}, \\
& + \bigcup_{n' \in \mathcal{N}_{DS}} \{(n', n_1)\}, \;\; \text{if } u \in P_1 \\
& + \bigcup_{n' \in \mathcal{N}_{RS}} \{(n'\,n_1), (n'\,n_2)\} \\
& + \bigcup_{n' \in \mathcal{N}_{UD}} \{(n_1\,n')\} \\
& + \bigcup_{n' \in \mathcal{N}_{LD}} \{(n_1\,n'), (n_2\,n')\} \\
& + \{(n_1\,n_2), (n_2\,n_1)\}.
\end{aligned}$$

3. *The row index sets for nodes n_1 and n_2 are the same as for node n: $\mathcal{R}_{n_1} = \mathcal{R}_{n_2} = \mathcal{R}_n$, and the column sets for new nodes are determined from the column partition: $C_{n_1} = P_1$ and $C_{n_2} = P_2$.*
4. *The arc types of the repositioned and duplicated arcs stay the same, and the two new arcs $(n_1\,n_2)$ and $(n_2\,n_1)$, become down and up ones respectively:*

$$\mathcal{T}_{n''n_1} \;=\; up, \forall n'' \in \mathcal{N}_{US}, (n''\,n_1) \in \mathcal{A}_{N+1},$$

$$
\begin{aligned}
\mathcal{T}_{n''n_1} &= down, \forall n'' \in \mathcal{N}_{DS_1}(n'' n_1) \in \mathcal{A}_{N+1}, \\
\mathcal{T}_{n''n_1} &= right, \forall n'' \in \mathcal{N}_{RS_1}(n'' n_1) \in \mathcal{A}_{N+1}. \\
\mathcal{T}_{n_1 n''} &= up, \forall n'' \in \mathcal{N}_{UD_1}(n_1 n'') \in \mathcal{A}_{N+1}, \\
\mathcal{T}_{n_1 n''} &= down, \forall n'' \in \mathcal{N}_{DD_1}(n_1 n'') \in \mathcal{A}_{N+1}, \\
\mathcal{T}_{n_1 n''} &= left, \forall n'' \in \mathcal{N}_{LD_1}(n_1 n'') \in \mathcal{A}_{N+1}, \\
\mathcal{T}_{n_1 n_2} &= down, \text{ and} \\
\mathcal{T}_{n_2 n_1} &= up.
\end{aligned}
$$

The main theorem of this thesis involves a generalized $\boxminus$ operation on nodes of networks in $\mathcal{G}^N$. We first prove the operation on a close cousin of the generic node, using node 3 in Figure 2.9.

Lemma 2.17 ($\boxminus$ on $\mathcal{G}^4$) *The $\boxminus$ operator as applied to the middle node in Figure 2.9 is a special case of the generalized $\boxminus$ operator on the generic node.*

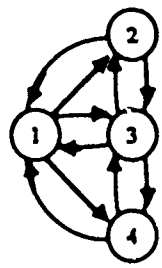

Figure 2.9: The 4-node generic network.

Proof: Proof by comparison. Take the case where the subsets of connected nodes each have one element, and $\mathcal{N}_{UD} = \mathcal{N}_{DS} = \{2\}$, $\mathcal{N}_{LD} = \mathcal{N}_{RS} = \{1\}$, $\mathcal{N}_{LS} = \mathcal{N}_{RD} = \{3\}$. We will call this network g_4. It has the following specification:

$$
\begin{aligned}
g_4 &= (\{1,2,3,4,\}, \\
&\quad \pi, \pi_1, \pi_2 \cup \pi_3, \pi_4, x_1, x_2, x_2, x_2,
\end{aligned}
$$

$\{(12),(21),(13),(31),(14),(41),(23),(32),(34),(43)\}$,
$\mathcal{T}_{12} = \mathcal{T}_{13} = \mathcal{T}_{14} = \text{right}, \mathcal{T}_{21} = \mathcal{T}_{31} = \mathcal{T}_{41} = \text{left}$,
$\mathcal{T}_{23} = \mathcal{T}_{34} = \text{down}, \mathcal{T}_{32} = \mathcal{T}_{43} = \text{up}$).

where $\pi = \pi_1 \cup \pi_2 \cup \pi_3 \cup \pi_4$. The operation on this network is $g_5 = g_4 \boxplus [\pi_2, \pi_3]$, where

$$
\begin{aligned}
g_5 \;=\; & (\{1,2,5,6,4,\}, \\
& \pi, \pi_1, \pi_2, \pi_3, \pi_4, x_1, x_2, x_2, x_2, x_2, \\
& \{(12),(21),(15),(51),(16),(61),(14),(41),(25),(52),(56),(65),(54),(45)\}, \\
& \mathcal{T}_{12} = \mathcal{T}_{15} = \mathcal{T}_{16} = \mathcal{T}_{14} = \text{right}, \mathcal{T}_{21} = \mathcal{T}_{51} = \mathcal{T}_{61} = \mathcal{T}_{41} = \text{left}, \\
& \mathcal{T}_{25} = \mathcal{T}_{56} = \mathcal{T}_{54} = \text{down}, \mathcal{T}_{52} = \mathcal{T}_{65} = \mathcal{T}_{45} = \text{up}),
\end{aligned}
$$

We enumerate the steps used to convert g_4 into g_5:

1. One node is discarded and two are added: $\mathcal{N}_2 = \mathcal{N}_1 - \{3\} + \{5,6\}$.

2. All arcs incident to node 3 are discarded. Of those, the horizontal ones are linked to nodes 5 and 6 according to the column partition, and the vertical ones are duplicated and incident to both nodes 5 and 6: $\mathcal{A}_2 = \mathcal{A}_1 - \{(13),(31),(23),(32),(34),(43)\} + \{(15),(51),(16),(61),(25),(52),(56),(65),(54),(45)\}$,

3. The row index sets for Nodes 5 and 6 are the same as for node 3: $\mathcal{R}_5 = \mathcal{R}_6 = \pi_2$, and the column sets for new nodes are determined from the column partition: $C_5 = x_2$ and $C_6 = x_3$.

4. The arc types of the repositioned and duplicated arcs stay the same, and the two new arcs (56) and (65) become down and up ones respectively.

∎

Theorem 2.18 ($\boxminus$ and $\boxplus$ on $\mathcal{G}^N$) *The $\boxminus$ operator and its dual $\boxplus$, as defined on the generic node, cover every possible transition of a network with N nodes to one with $N+1$ nodes.*

Proof: By induction:

1. Lemma ($\boxminus$ on $\mathcal{G}^4$).
2. Assume $\boxminus$ up to $\mathcal{G}^N$.
3. Demonstrate that $\mathcal{G}^N \boxminus [P_1, P_2] \rightarrow \mathcal{G}^{N+1}$ as follows:
   - Lemma ($\mathcal{G}^N \square \mathcal{N}^\bullet \rightarrow \mathcal{G}^4$);
   - Lemma ($\boxminus$ on $\mathcal{G}^4$);
   - the $\square$ operator is defined to be reversible, which implies that every node besides the two new ones n_1 and n_2, and every arc that was not incident to node n can be restored to its prior status as defined by g_N;

The dual argument holds by network duality. ∎

2.4 Summary

We arrived at the begining of this chapter carrying a transformation between linear programs and communication networks, and some node splitting operators.

- We proceeded to nest the operators and got: node ordering, cross splitting, and lots of duality through the choice of the first operation.
- The networks with three nodes were characterized in Table 2.1.
- The $\square$ operator was introduced and two lemmas about collapsing many nodes into one are proved.

- The generic node was introduced, and two lemmas followed. The first one showed that all nodes are similar to it or its dual. The second one showed that all networks can be reduced to some case in $\mathcal{G}^4$.

- The generalized forms of the $\boxminus$ and $\boxplus$ operators were introduced, and then shown to carry networks first from $\mathcal{G}^4$ to $\mathcal{G}^5$, and then from $\mathcal{G}^N$ to $\mathcal{G}^{N+1}$.

Finally, Theorem 2.18 acts as a characteristic mapping from one network to those having more nodes. Sometimes there is more than one path can be taken to the same network (associativity), and in addition, the inverse operation need not retrace the actual path taken to create the network. In the next chapter we will explore the transformation of networks into subproblems, and consult a parallel oracle.

Chapter 3

Parallel Decomposition

ALMOST daily, researchers in the technical disciplines envisage new and different uses for parallel computers. Linear programming as a practical field could never have happened were it not for the invention of the serial computer [Dan87], which revolutionized the approach to complex problems. And now, the availability of parallel computers will permit the next quantum expansion in the set of problems that can be solved. The parallel decomposition algorithm will be a first step in placing mathematical programming in league with other technologies making use of these new computers.

We view the ultimate information content of a problem formulation as the solution to the problem. To obtain the solution, we consult an oracle:

$$\text{solution} = \mathcal{O}(\text{ problem }).$$

Linear program solutions consist of points and/or rays of the primal and dual feasible regions of the problem. A typical oracle for solving linear programs is the simplex method:

$$\text{LP solution} = \text{simplex(LP)}.$$

This thesis is concerned with substituting various decomposition algorithms for the simplex method. The decomposition algorithms are governed by a communication

network between LP subproblems. Different problem structures will result in different networks and different subproblems. However, it is possible to define a single general algorithm having the communication network and the subproblem formulations as its parameters:

LP solution = decomposition (network, subproblems).

The goal of this chapter is to take the information contained in an LP and a communication network, to produce an equivalent set of information in the form of a system of subproblems and finally to find the LP solution using a parallel oracle operating on this equivalent information.

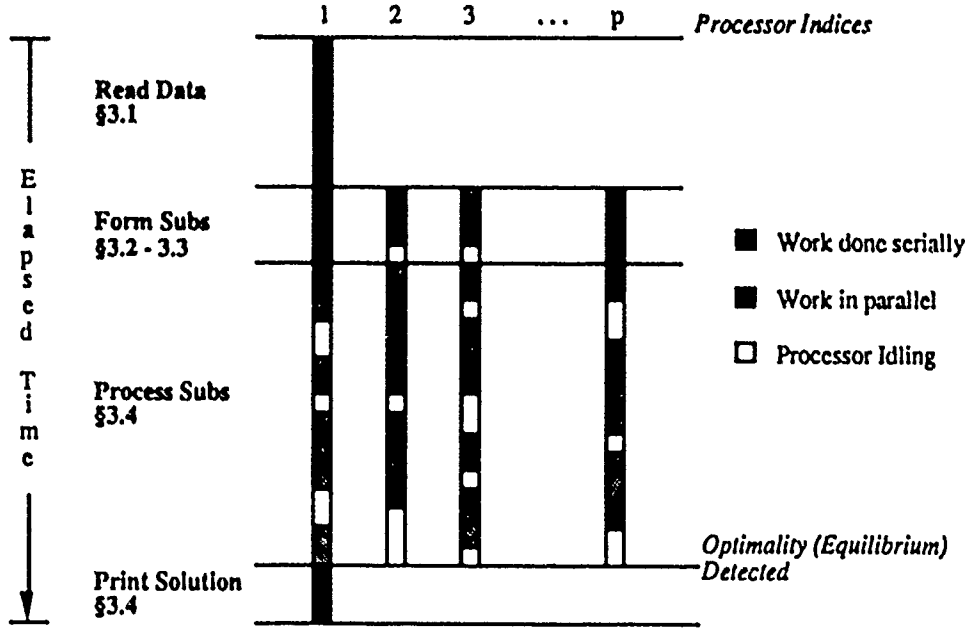

Figure 3.1: Strings of work.

Figure 3.1 lists the steps of parallel decomposition. Along with each step in the figure are the section numbers of this chapter that explain the step, and a representation of whether the step is done in serial or in parallel.

Read Data: The first order of business is to define the problem we wish to solve and the decomposition scheme used to do it. These are specified by the original data (A, b, c) and the communication network g.

Form Subs: The reverse transform from the communication network into a system of subproblems is covered in Sections 3.2-3.3. The information obtained in the Read Data step is processed in parallel during this step.

Process Subs: The parallel processors act as information carriers over the network, performing oracles on subproblems and filtering the solutions through interfaces. A relaxation of the nested oracle procedure is shown to perform $\mathcal{O}(1.1)$.

Print Solution: From the multitude of final subproblem oracles, we must construct one for $\mathcal{O}(1.1)$. Because the subproblem formulations contain all of the relevant subproblem solutions, this is a simple filtering process and is done serially.

3.1 Starting Information

In the following discussion, we will assume that our linear program formulation takes the form given in (1.1), namely

$$\begin{array}{lll} \min\limits_{x \geq 0} & & c^T x = z \\ \text{s.t.} & \pi: & Ax \geq b. \end{array} \tag{3.1}$$

3.1.1 The Problem Description

We can break down a problem description into two sets of information: the implicit information (indices and variables) and the explicit information (problem data).

Indices will play an important role in the discussions of problem structure and communicated information. Not only the constraint and variable indices are used,

but those for the right-hand side and objective as well. Thus, we define for LP (3.1):

σ to be the objective index,
s to be the right-hand side index,
$\mathcal{R}$ to be the row index set, and
$\mathcal{C}$ to be the column index set.

As in the previous chapter, when discussing partitions, we will use the same symbols for the *names* of the primal variables as for the *index sets* of their corresponding columns, and the same symbols for the *names* of the dual variables as for the *index sets* of their corresponding rows. Therefore, $\mathcal{R} = \pi$ and $\mathcal{C} = x$ for LP (3.1).

The *values* of the variables lie in vector spaces that are dimensioned in terms of their index sets. We see for LP (3.1) that

x the primal variables lie in $\Re^{\mathcal{C}}$, and
π the dual variables lie in $\Re^{\mathcal{R}}$.

Finally, the explicit information needed to give substance to the implicit information above is the problem data. We will take the convention of positioning this data within the problem by specifying its indices. For instance for (3.1):

A the constraint matrix is indexed by (π, x),
b the right-hand side vector is indexed by (π, s), and
c the cost vector is indexed by (σ, x),

This completes the specification of a linear programing problem

3.1.2 The Communication Network Description

The previous chapter explained the process of partitioning the row and column index sets. It also showed how communication networks result from this process. Rather than operating from partition information, we shall assume that the decomposition information is in the form of a communication network.

We repeat the definition from Chapter One for completeness. The communication network is the five-tuple,

$$g = (\mathcal{N}, \mathcal{R}_n, \mathcal{C}_n, \mathcal{A}, \mathcal{T}_a), \quad \forall n \in \mathcal{N}, a \in \mathcal{A},$$

where the tuples are defined to be

- $\mathcal{N}$ set of nodes,
- $\mathcal{R}_n$ node n's row index set,
- $\mathcal{C}_n$ node n's column index set,
- $\mathcal{A}$ set of arcs, $\mathcal{A} = \{(n_1\, n_2) \in \mathcal{N}^2 : \text{there is an arc from } n_1 \text{ to } n_2\}$,
- $\mathcal{T}_a$ the type for arc a (up, down, left, or right).

This completes the description of the decomposition scheme to solve LP (3.1).

3.2 Intermediate Information

Several information structures are constructed from the starting information in order to facilitate the formulations of the subproblems. These are: the *Incidence Graph* used to derive subproblem interfaces, the *Arc Index Sets* which index passed information, and the *Partition Graphs* which identify implicit subproblems and synchronized information.

3.2.1 The Incidence Graph

$$(A, b, c) \rightarrow h$$

The incidence graph h is created from the explicit information (A, b, c). It is bipartite with one class of nodes over the objective and constraint indices and the other class of nodes over the right-hand side and variable indices. Two nodes are connected (always between the two classes), if there is a nonzero entry in the data (A, b, c),

corresponding to the two indices of the linked nodes. For instance, if

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \pi = \begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix}, \quad A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ 0 \end{pmatrix}, \quad \text{and } c = \begin{pmatrix} c_1 \\ 0 \end{pmatrix}$$

then the corresponding incidence graph is that in Figure 3.2, and

$$h = (\{\sigma, \pi_1, \pi_2, s, x_1, x_2\}, \{(\sigma\, x_1), (\pi_1\, s), (\pi_1\, x_1), (\pi_2\, x_1), (\pi_2\, x_2)\}).$$

The nodes of h correspond to aggregations of the rows and columns of the linear program so that it represents the incidence between blocks of coefficients. Note that there is never a link between the two nodes σ and s, but all other links between the two classes of nodes are possible.

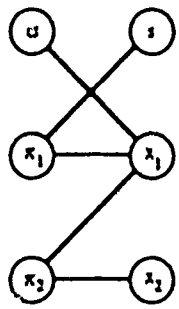

Figure 3.2: An incidence graph.

Future research along these lines will probably concern various optimal partitioning schemes, based on the coupling between subproblems and the level of computation needed to obtain subproblem solutions. Some good references on incidence graphs are [Ros70, Bun76, Tar76].

3.2.2 Arc Index Sets

$$(g, h) \rightarrow (\mathcal{R}_a, \mathcal{C}_a)$$

Recall from the Subproblem Interface Theorem that we need only pass a selection of a subproblem's solution over any given arc. The selection is made the arc's index set. We represent these sets as follows:

- if a is a vertical arc (up or down), it has a row coupling index set $\mathcal{R}_a$, and
- if a is a horizontal arc (left or right), it has a column coupling index set $\mathcal{C}_a$.

This means that either rows couple partitioned columns or columns couple partitioned rows. An arc is represented by two nodes. Thus arc index sets will have two nodes as subscripts. Node index sets have only one node for a subscript.

Let the arc $(n_1\, n_2)$ be horizontal; then

$$\mathcal{R}_{n_1 n_2} = \{k : \forall \text{ paths } jkl \text{ in the graph } h, k \in (\mathcal{R}_{n_1} \cap \mathcal{R}_{n_2}), j \in \mathcal{C}_{n_1}, l \in \mathcal{C}_{n_2}\}.$$

Let the arc $(n_1\, n_2)$ be vertical; then

$$\mathcal{C}_{n_1 n_2} = \{j : \forall \text{ paths } ijk \text{ in the graph } h, j \in (\mathcal{C}_{n_1} \cap \mathcal{C}_{n_2}) \cup s, i \in \mathcal{R}_{n_1}, k \in \mathcal{R}_{n_2}\}.$$

Secondly, the theorem says that according to given interfaces, the objective (right-hand side) values must appear in the topmost (leftmost) subproblems containing constrained variables (non-vacuous constraints), respectively. When a topmost subproblem has unconstrained variables or a leftmost subproblem has vacuous constraints, the theorem also says that an incoming up or left arc, respectively, carries the value of $c^T\tilde{x}$ or $\tilde{\pi}^T b$, respectively.

To determine in general whether an up arc ought to carry $c^T\tilde{x}$ along with $\tilde{x}$ and whether a left arc ought to carry $\tilde{\pi}^T b$ along with $\tilde{\pi}$, follow the simple rules:

1. up arcs carry $c^T\tilde{x}$ if there are objective coefficients in the formulation of a subproblem below, and
2. left arcs carry $\tilde{\pi}^T b$ if there are right-hand side coefficients included in the formulation of a subproblem to the right.

In other words:

$$\text{if } \mathcal{T}_{n_1 n_2} = \text{up, and } \exists n_3 \leq n_2 \text{ s.t. } c_{n_3}(j) = c_j \text{ for some } j \in \mathcal{C}_{n_3}$$

$$\text{then } \mathcal{C}_{n_1 n_2} \leftarrow \mathcal{C}_{n_1 n_2} \cup s.$$

Left arcs will carry $\tilde{0}$ if there are right-hand side coefficients included in the formulation of a subproblem to the right:

$$\text{if } \mathcal{T}_{n_1 n_2} = \text{left, and } \exists n_3 \leq n_2 \text{ s.t. } b_{n_3}(i) = b_i \text{ for some } i \in \mathcal{R}_{n_3}$$

$$\text{then } \mathcal{R}_{n_1 n_2} \leftarrow \mathcal{R}_{n_1 n_2} \cup \sigma.$$

3.2.3 Partition Graph Description

Partition graphs identify implicit subproblems created by cross-nesting the decomposition operators. They are used to designate what information must be *synchronized* by determining how λ, θ, l, and t are indexed. The solutions of subproblems corresponding to the nodes in a partition graph define the solution to an implicit subproblem. One that is not solved explicitly because it was decomposed. Before formally introducing the partition graph, we first define the following three graphs:

a vertical graph is a graph with all vertical arcs,

a horizontal graph is a graph with all horizontal arcs, and

a subgraph is a graph $s = (\mathcal{A}_s, \mathcal{N}_s)$, written $s \subseteq g$ where $g = (\mathcal{A}, \mathcal{N})$, if and only if $\mathcal{N}_s \subseteq \mathcal{N}$ and $\mathcal{A}_s \subseteq \mathcal{A} \cap \mathcal{N}_s^2$.

The information contained in the communication network g is used to generate its set of partition graphs $\mathcal{P}_g$ and their accompanying row and column index sets $\mathcal{R}_p$ and $\mathcal{C}_p$, for all $p \in \mathcal{P}_g$:

$$g \rightarrow (\mathcal{P}_g, \mathcal{R}_p, \mathcal{C}_p) \quad \forall n \in \mathcal{N}, a \in \mathcal{A}, p \in \mathcal{P}_g,$$

where

- $\mathcal{P}_g$ is the set of all partition graphs in g,
- $\mathcal{R}_p$ is the row index set for partition graph $p \in \mathcal{P}_g$, and
- $\mathcal{C}_p$ is the column index set for partition graph $p \in \mathcal{P}_g$.

Definition 3.1 (Partition Graph) *A partition graph p is a horizontal or vertical subgraph of g created by cross nesting one operator within another. The partition graphs of the communication network g are contained in the set $\mathcal{P}_g$. Each graph p is a collection of nodes $\mathcal{N}_p$ connected by arcs $\mathcal{A}_p$.*

For all $p = (\mathcal{N}_p, \mathcal{A}_p) \in \mathcal{P}_g$ we define row and column index sets to be the union of the node index sets contained in the graph:

$$\mathcal{R}_p = \bigcup_{n \in \mathcal{N}_p} \mathcal{R}_n, \quad \mathcal{C}_p = \bigcup_{n \in \mathcal{N}_p} \mathcal{C}_n.$$

Lemma 3.2 (Partition Graph Ordering) *There is a partial ordering on the partition graphs $\mathcal{P}_g$ of a given network $g \in \mathcal{G}^N$, based on the highest ordered node contained within them.*

Proof: There is an ordering on the nodes, and all partition graphs are maximally connected on horizontal or vertical arcs. Therefore, no partition graph can be a subgraph of another, and there must be a node in each that is of greatest order. Such nodes from different partition graphs are different and in turn partially ordered. ∎

Here are two properties of partition graphs.

Property 3.3 (Similar Rows or Columns) *If p is a horizontal partition graph, $\mathcal{R}_n$ is constant for all $n \in \mathcal{N}_p$. Likewise, if p is a vertical partition graph, $\mathcal{C}_n$ is constant for all $n \in \mathcal{N}_p$.*

Property 3.4 (Parent/Child Incidence) *If p and c are partition graphs and p is the parent of c, then if p is vertical, $\mathcal{C}_p = \mathcal{C}_c$ and $\mathcal{R}_c = \mathcal{R}_n$, where $n = \mathcal{N}_p \cap \mathcal{N}_c$. Likewise, if p is horizontal, $\mathcal{R}_p = \mathcal{R}_c$ and $\mathcal{C}_c = \mathcal{C}_n$, where $n = \mathcal{N}_p \cap \mathcal{N}_c$.*

Take as an example, the application of ⊞ on the bottom node of the D-W network g_D. The two partition graphs from that network are displayed in Figure 3.3. Their row and column index sets are

$$R_{p_1} = \pi_1 \cup \pi_2, \quad R_{p_2} = \pi_2,$$

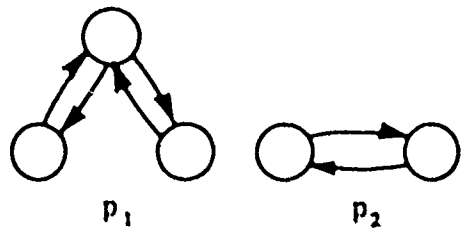

Figure 3.3: Partition graphs from splitting the bottom node.

$$C_{p_1} = x, \quad C_{p_2} = x.$$

The information carried by the up arcs from p_2 to node 1 must be synchronized and added to (2.9) as a single column. When formulating this subproblem, we purposely included a single set of variables l with which to take combinations of these columns.

3.3 Forming Subproblems

We concern ourselves now with a philosophical question on the information contained in a linear program specification, and how to obtain that information from a communication network in order to fully specify the subproblems used in a parallel oracle.

The following discussion concerns the dichotomy of structure and content. Translated to mathematics, this terms become symbols and meaning.

Definition 3.5 (Symbolic Representation) *An object is represented symbolicelly by the members of its structure and their relations to each other.*

Lemma 3.6 (Symbolic Linear Program) *A symbolic representation of a linear program is contained in $\mathcal{R}$ and C and an assumed standard form (3.1).*

Since subproblem n a linear program, its symbolic information consists of $\bar{\mathcal{R}}_n$ and $\bar{C}_n$ and an assumed standard form.

Theorem 3.7 (Necessary Information) *The following information is required to obtain a solution to a linear program: a* symbolic representation *in the form of*

row and column indices $\mathcal{R}$ and C for the default formulation (3.1), problem data *in the form of (A, b, c) for (3.1), and* an oracle.

To define the reverse transform, the indices of each linear program subproblem are obtained from the communication network by identifying their names and subscripts, and determining their dimensions. The result is a symbolic representation of each subproblem. Together with a description of the problem data and the simplex method, we can perform the oracle on any subproblem.

Definition 3.8 (Symbolic Subproblems) *The reverse transformation is an extraction of the symbolic representation of the subproblems from the communication network. We define it as a two step process:*

Index Sets *The subproblem index sets are defined in Tables 3.1 and 3.3 as a translation from the node index sets and the arcs entering the node. Each index has two parameters: its subscript, and its dimension.*

Default Formulation *Tables 3.2, 3.4, 3.5, and 3.6, comprise the standard subproblem formulation, defined in terms of the incidence between the subproblem's row and column indices. The standard subproblem formulation is summarized in Table 3.7.*

3.3.1 The Formulation Procedure

We follow a procedure of determining the subproblem index sets, which then determines the default formulation. From the position of a subproblem's corresponding node in the communication network (e.g. topmost or leftmost etc.) we can determine the partition of the original data over the set of subproblems.

Original Variables: The variables x_n and π_n appear in a subproblem based on the node index sets. If $\mathcal{R}_n$ is not empty then π_n appears. If C_n is not empty then x_n appears. It is possible for one to appear and not the other. These results are summarized in Table 3.1.

	Dimension	Subscript	Appears
π	$\mathcal{R}_n$	n	if $\mathcal{R}_n \neq \emptyset$
x	$\mathcal{C}_n$	n	if $\mathcal{C}_n \neq \emptyset$

Table 3.1: Original variables.

Original Data: Independent portions of A, b, and c will be used to define subproblems based on their node index sets:

$$A_n = \{A_{ij} : i \in \mathcal{R}_n, j \in \mathcal{C}_n\}.$$

Lemma 3.9 (Placement of b and c) *The right-hand side coefficients indexed by $i \in \mathcal{R}_n$ for node n are placed as follows:*

$$b_n(i) = \begin{cases} b_i & \text{if } n \text{ is maximal such that } (i\,j) \in \mathcal{A}_h \text{ for some } j \in \mathcal{C}_n, \\ 0 & \text{otherwise.} \end{cases}$$

The objective coefficients indexed by $j \in \mathcal{C}_n$ for node n are placed as follows:

$$c_n(j) = \begin{cases} c_j & \text{if } n \text{ is maximal such that } (i\,j) \in \mathcal{A}_h \text{ for some } i \in \mathcal{R}_n, \\ 0 & \text{otherwise.} \end{cases}$$

Proof:

1. Begin with full arc index sets and leftmost and topmost placement of b and c, respectively.

2. By the Subproblem Interface Theorem, we redefine the arc index sets of those down and right arcs $(n\,n')$ incident to topmost and leftmost nodes $n \in \mathcal{N}$ and thus move down all $c_n(j) : j \in \mathcal{C} - \mathcal{C}_{nn'}$, and right all $b_n(i) : i \in \mathcal{R} - \mathcal{R}_{nn'}$.

3. For each down arc, there is a corresponding up arc that must have its index set augmented by σ for the objective row if a node below contains any original objective coefficients.

4. Steps 2 and 3 can be repeated as often as necessary to achieve the result in Lemma 3.9.

∎

Original data are indexed by the node index sets. Both the data and their indices carry the same subscripts. These results are summarized in Table 3.2.

	Indices	Subscript	Appears
A	π, x	n	if $\mathcal{R}_n \times C_n \neq \emptyset$
b	π, s	n	if $\mathcal{R}_n \neq \emptyset$
c	σ, x	n	if $C_n \neq \emptyset$

Table 3.2: Original data.

Incoming Arcs: The added variables and added data of a subproblem are those other than the originals. Their appearance in the formulation is governed by the incoming information, i.e., the incoming arcs. They form structures for handling the information as it arrives, placing it into the formulation so that it will have the proper effect.

An ambiguity arises here. Information transported along up (left) arcs is used to form additional rows (columns) in the formulation. If there is more than one up or left arc, there can be a choice as to how the information gets incorporated into the formulation that is not specified in the communication network. That choice, for adding columns, has to do with the number of convexity constraints to keep. One is sufficient, but more than one will give the region being approximated greater resolution. Our default choice will be for the latter.

When the incoming up arc has its source in a different partition graph, there is no choice; there must be one convexity constraint for each such partition graph. This forces the information from each partition graph to be coordinated into one new column. Likewise, for left arcs the default will be to add individual constraints, and

only when the arc's source lies in a different partition graph will we add only one constraint for all the information arriving from that graph.

In the following descriptions of added variables and data, we adopt the convention that the generic incoming arc to node n is $a = (n_1, n)$.

Added Variables: All added variables are subscripted by the incoming arc that generated them, except for the case when the source of the incoming arc is in a child partition p. The variables λ, θ, l, and t should then be subscripted by p. This causes a single primal or dual convexity constraint to be created for each child partition as required. Table 3.3 shows which added variables are affected by synchronization.

The dimensions K_{n_1} and K_p are defined as

K_n : the number of solutions broadcast by subproblem $n \in \mathcal{N}$,
K_p : the number of solutions broadcast by partition graph $p \in \mathcal{P}$,

where the term *broadcast* refers to the practice of communicating a subproblem solution over the outgoing arcs of the corresponding node.

	subscript	appears when incoming	Dimension
λ	a or p	left arc	K_{n_1} or K_p
υ	a	down arc	1
ψ	a	up arc	C_a
ω	a	right arc	1
θ	a or p	up arc	1
l	a or p	up arc	K_{n_1} or K_p
u	a	right arc	1
y	a	left arc	$\mathcal{R}_a$
w	a	down arc	1
t	a or p	left arc	1

Table 3.3: Added variables.

Added Data: These structures are generated by incoming arcs. Information carried by up and left arcs is *accumulated*, whereas that carried by down and right arcs is *overwritten*. This important point separates standard LP decomposition from the class of totally symmetric algorithms. One method proposed to overcome this is to incorporate a proximal-point penalty term into the objective function [Roc76, Gol86, BeTS9]. Table 3.4 shows which data are affected by synchronization. The indexing of each data item positions it with respect to the subproblem variables and constraints. When an added variable is subscripted by an incoming arc, the incident data is subscripted by the *reverse arc*. The reverse of arc $a = (n_1, n)$ is $\bar{a} = (n, n_1)$. When an added variable is subscripted by a partition graph p, the incident data $\tilde{\gamma}$ and $\tilde{y}$, are subscripted by p also. The incident data $\tilde{\Pi}$ and $\tilde{X}$ are subscripted by the incoming arc. The indexing then defines a single block of constraints or columns, since one is subscripted by the arcs and the other by p.

We now offer word descriptions of the added data presented in Table 3.4:

$\tilde{\gamma}_{\bar{a}}$: the optimality indicators for the dual solutions that are passed over arc a,

$\tilde{y}_{\bar{a}}$: the optimality indicators for the primal solutions that are passed over arc a,

$\tilde{\Pi}_{\bar{a}}$: the *translated* dual solutions that are passed over arc a,

$\tilde{X}_{\bar{a}}$: the *translated* primal solutions that are passed over arc a,

$I_{\bar{a}}$: a matrix that translates dual (primal) solutions passed over up (left) arc a.

The passed information is either placed directly into the formulation of the destination subproblem, or incorporated into an existing structure. For down and right arcs, only the latest solution is used. The new information is written directly over the old and appears in the formulation as $\tilde{\psi}_{\bar{a}}$ or $\tilde{y}_{\bar{a}}$. For up and left arcs, the information is *accumulated,* and appears as an expandable structure in the formulation of the destination subproblem. Each new piece of information causes the row or the column dimension of the structure to increase by one, and so these dimensions are indexed by the number of times the source subproblem has been solved. Specifically, for $k \in K_n$

	Subscript	incoming	Indexing
1	none	down arc	θ, s
$\tilde{g}$	$\bar{a}$ or c	up arc	θ, l
$\tilde{X}$	$\bar{a}$	up arc	ψ, l
$-I$	$\bar{a}$	up arc	ψ, x
$\tilde{\psi}$	$\bar{a}$	down arc	v, x
$\tilde{\theta}$	$\bar{a}$	down arc	v, s
$\tilde{\delta}$	$\bar{a}$	down arc	v, w
$\tilde{\delta}$	$\bar{a}$	down arc	σ, w
1	none	down arc	σ, t
$\tilde{\gamma}$	$\bar{a}$ or c	left arc	λ, t
$\tilde{\Pi}$	$\bar{a}$	left arc	λ, y
$-I$	$\bar{a}$	left arc	π, y
$\tilde{y}$	$\bar{a}$	right arc	π, u
$\tilde{t}$	$\bar{a}$	right arc	σ, u
$\tilde{d}$	$\bar{a}$	right arc	ω, u
$\tilde{d}$	$\bar{a}$	right arc	ω, s

Table 3.4: Added data.

and arc $a = (n_1, n)$,

$$\tilde{\gamma}_a^k = \tilde{g}_a^k = \begin{cases} 1 & k^{\text{th}} \text{ solution to } n_1 \text{ is optimal,} \\ 0 & \text{otherwise,} \end{cases}$$
$$\tilde{X}_a^k = \{\tilde{x}_{n_1 j}^k : j \in \mathcal{C}_a\},$$
$$\tilde{\Pi}_a^k = \{\tilde{\pi}_{n_1 i}^k : i \in \mathcal{R}_a\}.$$

The following matrices are permutation matrices (not necessarily square). Their entries serve to translate the primal (dual) solution of an up (left) arc into the rows (columns) of the destination that are coupled to the source. For a given vertical arc $a = (n_1, n)$,

$$I_a^{ij} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ element of the set } \mathcal{R}_a \text{ is } i, \\ 0 & \text{otherwise.} \end{cases}$$

If a is horizontal, then

$$I_a^{ij} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ element of the set } \mathcal{C}_a \text{ is } j, \\ 0 & \text{otherwise.} \end{cases}$$

Non-negativity: Variables restricted to be non-negative and others not sign restricted are given in Table 3.5. The original variables x_n can be either non-negative or free. The default is free. When n is a bottommost node, $x \geq 0$ with a non-vacuous original column. It is true that x_n could be non-negative in subproblems that are not bottommost, but it is sufficient that the condition hold in any one subproblem. We chose the bottommost one to guarantee that it has at least one extreme point.

Index	Setting
l_a	≥ 0
u_a	≥ 0
x_n	free or ≥ 0
y_a	free
w_a	free
t_a	free

Table 3.5: Non-negativity.

Constraint Types: The types for constraints, whether equality or inequality are given in Table 3.6. There are two choices for the π_n corresponding to the primal constraints. The default is equality. If n is a rightmost node, the constraint is an inequality as shown. Similar to determining non-negativity for x_n, all π_n constraints could be inequalities but it is sufficient for only those in rightmost subproblems with non-vacuous original constraints.

Index	Setting
λ_a, λ_p	$\geq$
v_a	$\geq$
π_n	$=$ or $\geq$
ψ_a	$=$
ω_a	$\geq$
θ_a, θ_p	$=$

Table 3.6: Constraint types.

3.3.2 Summary

The variable and data information tables are partially summarized in Table 3.7. Subproblem formulations are derived from this standard form. Given a node n and all its incoming arcs, e.g., $a = (n_1, n)$, this table will generate one subproblem in the schema of the communication network.

The most interesting feature of Table 3.7 is its symmetry with respect to the relations between Dantzig-Wolfe and Benders decomposition. The Greek and Roman symbols are interchanged by taking the transpose. Another feature to notice is that the series of entries $\tilde{X}_{\bar{a}}$, $\tilde{y}_{\bar{a}}$, $\tilde{\psi}_{\bar{a}}$, and $\tilde{\Pi}_{\bar{a}}$ are all added data structures to handle passed information with entries in real space, while the series of entries $\tilde{g}_{\bar{a}}$, $\tilde{d}_{\bar{a}}$, $-I_{\bar{a}}$, $-I_{\bar{a}}$, $\tilde{\delta}_{\bar{a}}$, $\tilde{\gamma}_{\bar{a}}$ are added structures with entries in binary space. The entries in the second series serve as indicators of what functions their corresponding real space information will serve, and how they will impact the subproblem formulation. Together, both

	l_a	u_a	x_n	y_a	w_a	t_a		s_n
λ_a				$\bar{\Pi}_{\bar{a}}$		$\bar{\gamma}_{\bar{a}}$	$=$	0
v_a			$\bar{\psi}_{\bar{a}}^T$		$\bar{\delta}_{\bar{a}}$		$\geq$	$-\bar{\theta}_{\bar{a}}$
π_n		$\bar{y}_{\bar{a}}$	A_n	$-I_{\bar{a}}$			$=/\geq$	b_n
ψ_a	$\bar{X}_{\bar{a}}$		$-I_{\bar{a}}$				$=$	0
ω_a		$\bar{d}_{\bar{a}}$					$=$	$\bar{d}_{\bar{a}}$
θ_a	$\bar{g}_{\bar{a}}^T$						$=$	1
	≥ 0	≥ 0	$free/\geq 0$	$free$	$free$	$free$		
σ_n	0	$-\bar{l}_{\bar{a}}$	c_n	0	$\bar{\delta}_{\bar{a}}$	1		

Table 3.7: Template for generating subproblems.

series are a constellation of structures bordering the original data A_n like the planets orbiting the sun, each bringing to bear its own fundamental force on the central mass.

3.4 The Parallel Oracle

In this section we assume that we have available to use a collection of decomposition subproblems that are an equivalent symbolic representation of some original LP formulation. When coupled with data values and an oracle we can obtain a solution to the original LP.

When we finish solving a subproblem in a decomposition scheme it is well known that any neighboring subproblem on the network is now eligible to receive the solution for the purpose of updating its formulation. The discussion that follows comes from a very simple idea:

Why not solve all of the neighboring subproblems at the same time?

Thus, we will modify the nested oracle, which was designed to work between two problems (a Master and a Slave). There are two steps:

- enlarge the set of communicable information to include interior points of the Slave, but keep it finite, and

- broadcast information instead of having only two-way conversations,

where we intend the term *broadcast* to mean that a node communicates information over its outgoing arcs to all of its neighbors in a communication network. The first step completely blurs the distinction of Master and Slave, and the second suggests using a parallel computer. The proof of the oracle is in terms of the validity of the above-two relaxations of the nested oracle.

Definition 3.10 (Relaxed Oracle) *When consulted, the relaxed oracle $\mathcal{O}_r(\cdot)$ provides either:*

- *a primal feasible point $\dot{x}$ or a dual feasible point $\dot{\pi}$,*
- *a feasible primal ray $\vec{x}$, or*
- *a feasible dual ray $\vec{\pi}$,*

where this information is taken from a finite set that includes all extreme points and extreme rays.

Lemma 3.11 (Relaxed Oracle) *The finiteness argument for the D-W method is not inhibited by a substitution of the relaxed oracle $\mathcal{O}_r(\cdot)$ for the regular oracle $\mathcal{O}(\cdot)$ in Steps 2 and 3.*

Proof: A review of that argument will show that the information communicated up and to the left between subproblems has two essential properties:

- the information comes from a finite set;
- the finite set includes all extreme points and rays.

∎

Lemma 3.12 (Broadcasting Information) *The practice of broadcasting subproblem solutions does not inhibit finite convergence of the D-W method.*

Proof: The proof is simple. Broadcasting does not alter the set of communicable information when the relaxed oracle is used. ∎

The following is a corollary to the Reverse Transform Theorem that will be referred to for direction in the Parallel Oracle.

Corollary 3.13 (Subproblem Modifications) *Arc types govern the types of modifications made to their destination subproblems as follows:*

up arc *add a column,*
down arc *modify the objective function,*
left arc *add a row,*
right arc *modify the right-hand side.*

The Overall Solution Lemma tells how the solution of (3.1) is constructed from the individual subproblem solutions.

Lemma 3.14 (Overall Solution) *The primal and dual solutions $(\bar{x}, \bar{\pi})$ to (3.1) are*

$$\bar{\pi} = \bigcup_{n \in \mathcal{N}_L} \bar{\pi}_n, \text{ and } \bar{x} = \bigcup_{n \in \mathcal{N}_U} \bar{x}_n.$$

where $\mathcal{N}_L$ are the leftmost nodes of g and $\mathcal{N}_U$ are its topmost nodes.

Proof: By induction on the levels of partition graphs.

1. The lemma is true for any vertical or horizontal partition graph:
   - The lemma is true for the D-W and Benders Methods.
   - Assume the lemma is true for a partition graph with l levels.
   - Use D-W or Benders method on the rightmost or bottommost two subproblems of a partition graph with $l + 1$ levels and reduce the number of levels to l.
2. Assume the lemma is true for l levels of partition graphs.
3. If another level of partition graphs is added to the network, it will be to the right or below, leaving the solution still at the top and leftmost nodes. So the lemma must be true for $l + 1$ nodes also.

∎

The statement of the parallel oracle is based on the premise of independent work units we will call *jobs*. Our work units are modifying and solving subproblems, so there is a one-to-one correspondence between jobs and subproblems. Jobs are *submitted* to be serviced by any processor, and held *pending* until one becomes available. We use the term *non-pending* in the theorem to refer to those jobs that are not waiting to be processed; either running or not submitted.

Theorem 3.15 (Parallel Oracle) *This procedure performs* $\mathcal{O}(3.1)$*:*

1. *Formulate all of the subproblems* $\{1, \ldots, N\}$ *and submit a job for each one.*
2. *Repeat the following until there are no more jobs:*
   - *Get a job with its associated subproblem* n*.*
   - *Use the Subproblem Modification Lemma to determine what modifications to make to the subproblem based on any new information.*
   - *Consult the oracle* $\mathcal{O}(n)$*.*
   - *If the oracle does not repeat the same solution then broadcast it and submit a job for each non-pending neighbor.*

Proof: We need to show that solutions provided by $\mathcal{O}(\cdot)$ for any subproblem will satisfy the restrictions for information passed over arcs. These restrictions are defined by nesting the relaxed oracle $\mathcal{O}_r$. The proof is by induction on the number of levels of partition graphs.

1. Information passed up and left from one partition graph to another satisfies $\mathcal{O}_r(\cdot)$:
   - The oracle-provided solution to a D-W Master problem always satisfies the conditions for the relaxed oracle.
   - Assume that for vertical partition graphs with l levels that oracle provided solution of the topmost node satisfies the conditions for the relaxed oracle.

- Take a vertical partition graph with $l+1$ levels. The non-topmost nodes implicitly represent a D-W Slave problem and they themselves satisfy the relaxed oracle by the induction step. The topmost node must also satisfy the relaxed oracle by the Master/Slave relation and by the fact that a top node in a two-node scheme also satisfies the relaxed oracle.

Take the dual of the above argument to prove the case for left arcs.

2. Assume the lemma is true for l levels of partition graphs.

3. Take a network with $l+1$ levels of partition graphs, where the first one is vertical. The second-level graphs provide solutions to the first level that satisfy the relaxed oracle by the induction step. By Step 1, the first level must also, and thus all $l+1$ levels.

∎

This chapter has been an outline of how to implement parallel decomposition from start to finish. We assumed that the work of defining a communication network was already done, and that the remainder of the work was to form subproblems and execute the parallel oracle. One subtle point was made in the Overall Solution Lemma, and that is that it is a relatively simple matter to construct the overall solutions. By more traditional methods, this is often a tricky exercise in data management.

Finally, the simple loop of: *Listen, Modify, Evaluate, and Broadcast* is our gerbil on a treadmill, which together with many others like it, are more powerful than the strongest workhorse; and faster too. We will see this conclusion supported in the results of the next chapter.

Chapter 4

Results for Staircase Linear Programs

DOES parallel decomposition make effective use of the machine it is designed to exploit? A Fortran77 program which solves Staircase Linear Programs was written to find a practical answer to this question. This code has run on two different shared-memory multiprocessing computers: a Sequent Balance 8000, and an IBM 3090/600E. Preliminary results on the Sequent computer were reported in [Ent88]. More extensive results on the IBM 3090 will be reported here. The parallel algorithm is inherently message based. As a result, the shared-memory implementation actually simulates a message-passing/distributed-memory parallel computer, using the Intel iPSC subroutine library as a standard interface.

Naturally, the decomposition code must solve linear program subproblems. This is accomplished by calling MINOS 5.1 [MS87] as a subroutine [Ent87]. Likewise, the best comparison of the decomposition method is to solve the same test problems using MINOS as a stand-alone system. This approach allowed many implementation differences to be eliminated, and permitted the merits of decomposition and parallel decomposition alone to be discussed. The tests of the decomposition code were designed to:

- produce results from which to judge the merits of parallel decomposition,
- investigate the algorithm's performance under different parameter settings,
- provide performance extrapolations outside the set of test problems, and
- outline the current limitations of the code and areas for improvement.

Emphasis is placed on demonstrating that decomposition and added processors provide faster solutions, with acceptable accuracy.

Our presentation of computational results is based on the suggested standards of [CDM79] and [JBNP89]. In addition, several similar presentations were considered, including [HieS2, HLS1b]. Section 4.1 covers the theoretical basis of the computer code, and its software implementation. Section 4.2 gives details of the experimental apparatus and presents results that support the appropriateness of parallel decomposition on staircase problems. Finally, the conclusions section argues the case for parallel decomposition in general and traces directions for future work in the field.

4.1 General Information

Method: Staircase subproblems were formed and solved on an "as available" basis using p processors. Subproblems are considered available when they have just received new information from an adjacent node on the network. When a subproblem finishes optimal, both the primal and dual solutions are communicated. When infeasible, only the dual solution is broadcast, and when unbounded, only the primal solution is broadcast. No dual optimal solution can be sent until one has been received, except for rightmost subproblems. As a result, the Phase I algorithm (for obtaining a primal feasible solution) is a serial one. Computational results exhibit this property. Also, the results show parallel decomposition outperforming the simplex method on problems having more than 2000 nonzero entries.

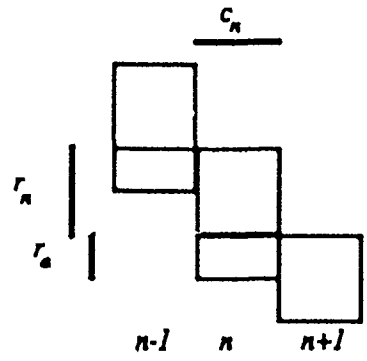

Figure 4.1: Dimensions of a step.

Memory Space: The size of the Fortran code and the amount of memory required for data storage are parameterized in the following terms:

$N = \#$ subproblems (max N is $\hat{N} = 51$),
$p = \#$ processors (max p is $\hat{p} = 20$),
$r_a = \#$ coupling rows for arc a (max r_a is $\hat{r}_a = 300$),
$r_n = \#$ nonzero rows in n's partition of $A - r_{(n,n+1)}$,
$c_n = \#$ columns in n's partition of A,
$e_n = \#$ nonzeros in n's partition of A,
$\bar{r}_n = \#$ rows in subproblem n,
$\bar{c}_n = \#$ columns in subproblem n, and
$\bar{e}_n = \#$ nonzeros in subproblem n.

The maximum values for N, p and r_a are given for the test configuration. Figure 4.1 represents the pattern of nonzero coefficients near the nth partition. The lengths of the bold lines show the dimensions of r_n, $r_{n,n+1}$, and c_n for this partition. Closed-form equations for the subproblem dimensions $(\bar{r}_n, \bar{c}_n, \bar{e}_n)$ are

$$\begin{aligned} \bar{r}_n &= r_n + r_{n,n-1} + r_{n,n+1}, \\ \bar{c}_n &= c_n + r_{n,n+1}, \\ \bar{e}_n &= e_n + r_{n,n+1}^2 + 2r_{n,n-1}. \end{aligned}$$

Given the values of $N, \bar{r}_n, \bar{c}_n, \bar{e}_n$, an expression for the total amount of shared memory used by the program can be calculated as

$$\# \text{ bytes } = 496\hat{N} + 16\hat{p} + 16\hat{N}\hat{r}_a + 8\sum_n(\bar{e}_n(1.25 + 4.5\bar{r}_n/\bar{c}_n) + 17.5\bar{r}_n + 6.75\bar{c}_n).$$

Software: The computer code presented here is based on MINOS 5.1. It uses MINOS in its entirety, with a few extra routines spliced in here and there. MINOS consists of three basic modules: *Input*, *Solve*, and *Output*. The parallel decomposition algorithm has two additional modules. The first, *Form Subs*, is inserted after the MINOS Input module, and the second, *Process Subs*, governs parallel MINOS Solves and decomposition message handling. In addition, a small amount of extra work is involved in collating the many subproblem solutions into one overall solution before they are Output. Thus, the Input/Output work is slightly greater for decomposition.

MPS and SPECS Input Files: These files are input using the MINOS Input Module. The standard MPS file is input to determine the Problem Data. It is assumed that this MPS file describes an LP that has a block diagonal or staircase structure. Normal MINOS input files are sufficient to solve the LP as a single large problem. However, to decompose a block diagonal structure into n subproblems, additional information must be provided in the DSPECS file.

DSPECS Input File: This file contains the *additional* information needed to complete a staircase decomposition linear program. An example of such a file is:

0		Debugging Parameter
50		% of extra rows to add to each subproblem
100		% of extra columns to add to each subproblem
3		number of subproblems
20	30	number of rows and columns in the first subproblem (optional)
20	30	number of rows and columns in the second subproblem (optional)
20	30	number of rows and columns in the third subproblem (optional)
4		number of processors (actually specified in JCL)

Since this is strictly Benders decomposition on a staircase system, the number of subproblems N equals the number of nodes, and $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{A} = \{(1,2),(2,1),\ldots,(N-1,N),(N,N-1)\}$, $\mathcal{T}_a$ = left, if $a = (n, n-1)$ for some $n \in \mathcal{N}$, and $\mathcal{T}_a$ = right, if $a = (n-1, n)$ for some $n \in \mathcal{N}$. This means that the entire communication network is defined in terms of N and $|\mathcal{C}_n|$, $\forall n \in \mathcal{N}$. The sets $\mathcal{R}_n$ and $\mathcal{C}_n$ are well defined given the number of columns in each partition and the fact that the elements of $\mathcal{R}$ and $\mathcal{C}$ are ordered.

Output Files: Each processor has a standard Fortran output file, and therefore the MINOS-type iteration log of each subproblem solved by each processor will appear in the corresponding file. In addition, the root process appends decomposition and parallel computation summary statistics, and the overall LP solution in its standard output file. The solution has the same format as MINOS. Finally, one short summary file is written by the root process that also contains the summary statistics.

Forming Subproblems

The serial version of this module was first documented in [Ent87]. Chapter 3 described for the most general case, how to form subproblems. The staircase version implicitly assumes a specific communication graph as in Figure 4.2 and that no row or column

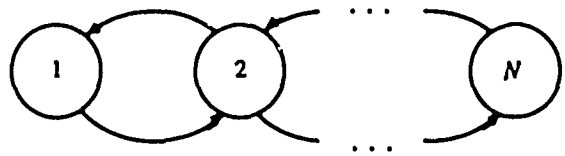

Figure 4.2: Linear communication network for staircase pattern.

permutations are necessary to obtain a staircase pattern in A. Before the *Form Subs* step, the subproblems are dimensioned based on these implicit assumptions. No intermediate information, as outlined in Chapter 3, need be used. A simple heuristic is used to provide subproblem dimensions. The object of the heuristic is to partition

the ordered columns of the matrix into a user-specified number of sets n. The columns in each set are adjacent, and they are chosen so as to minimize the number of coupling rows between the columns of adjacent sets. First a profile function $f(\cdot) : \mathcal{C} \rightarrow \mathcal{R}$ is calculated, where

$$f(j) = \max_{j<j'\leq|\mathcal{C}|}\{i \in \mathcal{R} : A_{ij'} \neq 0\} - \min_{1\leq j'\leq j}\{i \in \mathcal{R} : A_{ij'} \neq 0\}.$$

Then $N-1$ local minima are found so that the distances between them are nearly the same. It is also desirable to get the local minima as small as possible. If the problem has not enough steps to supply N subproblems, a warning is printed, and the number of steps found is used.

Processing Subproblems

When the solution of a neighboring subproblem arrives at the mailbox of a given subproblem, an independent job is defined. (Independence between jobs means that they can be executed concurrently.) Messages from different subproblems can be handled by a single job as described in Table 4.1.

Receive Message(s)
Make Modification(s)
Perform Oracle
Broadcast Solution
Die

Table 4.1: The life of a job.

Jobs are serviced, in the order they were made available, by any available processors. Messages from different subproblems can be handled by a single job as described in Table 4.1, which lists the four associated steps. The first round of jobs may skip the first two steps if there are no messages to receive.

Since this is an implementation of Benders decomposition, if a primal solution is received, the RHS is modified. If a dual solution is received, a new constraint is added.

The oracle is performed by a subroutine call to the Solve Module of MINOS. A pointer, passed as a parameter, directs MINOS to the proper data set, and in return, some solution form is provided according to the oracle's definition.

Different information is broadcast under the differing exit conditions of the Solve Module.

When optimal, the primal extreme point is passed over an outgoing right arc (if one exists), and the dual extreme point is passed over an outgoing left arc (if one exists) only if there is already an extreme point in the present subproblem's extra constraints, or it is rightmost. This guarantees dual feasibility of the information passed on left arcs.

n unbounded, in addition to the primal extreme ray, a primal extreme point adcast over an outgoing right arc (if it exists). Since MINOS is an implementation of the simplex method, the extreme point is available and used.

When infeasible, the dual extreme ray is passed left. In this situation, the parallel decomposition algorithm is actually serial, because only one new job is created from that finishing. Some test problems spend much of the time with infeasible subproblems. An extreme example is SC205, which has only a single nonzero objective coefficient in the leftmost subproblem. This makes all but the leftmost subproblem feasibility problems: we need only find a feasible point because the objective is vacuous. The decomposition algorithm can be made parallel by passing an infeasible primal solution to the right, but this information must not be relied on as part of an overall solution. At the time of this writing, we have yet to implement this feature.

Convergence and the Termination Criterion

Dual solutions are extreme points of the dual feasible region of the neighbor and therefore finite in number. If a dual solution corresponds to a non-binding constraint, the job is not executed. Eventually, no new jobs will be created; at this point, an optimal solution has been found.

To test whether a constraint will be binding, the objective value of the subproblem n_2 that sent the dual extreme point is compared with the value of the variable t_a in subproblem n_1 where $a = (n_2, n_1)$ is the arc that carried the message. Since t_a is a lower bound on the value of z_{n_2}, if

$$z_{n_2} - t_a < tol,$$

then the constraint will be non-binding. The value of *tol* is the default feasibility tolerance used by MINOS.

Discarding Constraints

Typically, a large number of constraints will be added to a given subproblem. However, not all of them are necessary to obtain an optimal solution. At most $|\mathcal{R}_a|$ can be binding at the final solution. We actually keep $|\mathcal{R}_a| + 2$ for good measure. The decomposition code overwrites the added constraints that are no longer binding. It replaces the constraint that has been slack for the greatest number of solves.

Communication

Messages contain a quantity of information that is a function of the number of coupling constraints r_a between the communicating subproblems. Table 4.2 gives the lengths of each message type in bytes. The maximum message length is $16*(3+86) = 1424$ bytes for all the test problems.

Sending a message involves loading it into a buffer and copying the buffer into the proper mailbox. Receiving a message involves copying it from the proper mailbox

Message	Bytes
Primal Point	$8 * (3 + r_a)$
Primal Point and Ray	$16 * (3 + r_a)$
Dual Point or Ray	$8 * (4 + r_a)$

Table 4.2: Message sizes.

into a buffer. Subproblems have one mailbox for each incoming arc. Each mailbox is capable of holding only one message. If a new message arrives before the old one is read, the old one is discarded. Discarding messages in this fashion does not affect finite convergence (but according to [HSL88], it is possible for such retained information to be used to speed convergence).

Basis Factorization

MINOS maintains a basis factorization that is updated by the decomposition code as appropriate after each modification to a subproblem. The routines for this purpose are in the software package called LUSOL and are documented in [GMSW86]. As a result of making both row and column updates, the factorization needs to be recalculated only when it becomes inaccurate or too large. The default settings from MINOS are used to govern refactorization.

4.2 Testing

The following experiments were performed to test the performance of parallel decomposition algorithms. A test suite of twenty-two staircase linear programs were solved with different partitions and different numbers of processors. The conclusions are that the algorithm is consistently well behaved in its use of additional processors and that it outperforms the serial algorithm (the simplex method) in most cases, when using only four processors.

4.2.1 The Test Environment

We report the test environment so that the interested reader may reproduce the same conditions on a variety of parallel machines.

Language	Fortran 77 with IBM Parallel Fortran Extensions
Compiler	IBM Parallel VS Fortran with VS Fortran V2 Rel 1.1
Compiler Options	No Vector, No Parallel, Optimize Level 3, Dynamic Shared Common
Computer	IBM 3090/600E, 2Gbytes shared virtual memory, and 128Mbytes real extended memory.
Operating System	MVS/XA V2.2.0
Code + Local Common	0.62 Mbytes
Shared System Common	0.27 Mbytes
Shared Data Common	1.60 Mbytes
Total Shared Common	1.87 Mbytes
Total Memory	2.49 Mbytes
Tolerances	MINOS Defaults
Message Passing	w/o locks
Job Flow Control	with locks

The processors were aligned after dispatch with a barrier.

4.2.2 The Test Suite

All but three of the twenty-two staircase-linear-program test problems were chosen from a collection of fifty-three used by Lustig [Lus87] in a performance evaluation of the simplex method. Included in his report are pictures of the patterns of nonzeros for the test suite: see Appendix B.

Table 4.3 lists the LP dimensions for the test suite. The problems are ordered by the number of nonzeros. All but three are part of a set of test problems made available by Gay [Gay85] and distributed over *netlib* [DG87]. The DIET series of test

Prob. #	Prob Name	Rows	Cols	Elems	Obj. Value (netlib)	Netlib #
1	DIET2	5	12	48	1.8500000000000E+02	N/A
2	DIET3	8	18	72	2.7750000000000E+02	N/A
3	DIET7	15	42	168	6.4750000000000E+02	N/A
4	SC205	206	203	552	-5.2202061211707E+01	15f
5	SCAGR7	130	140	553	-2.3313892547843E+06	17f
6	SCORPION	389	358	1744	1.8781248227381E+03	21f
7	SCAGR25	472	500	2029	-1.4753433060769E+07	16f
8	SCTAP1	301	480	2052	1.4122500000000E+03	26f
9	SCFXM1	331	457	2612	1.8416759028349E+04	18f
10	GROW7	141	301	2633	-4.7787811814712E+07	8f
11	SCSD1	78	760	3148	8.6666666743334E+00	23f
12	STAIR	357	467	3857	-2.5126695119000E+02	11
13	SCRS8	491	1169	4029	9.0429695380079E+02	22f
14	PILOT4	411	1000	5145	-2.5810162253381E+03	11f
15	SCFXM2	661	914	5229	3.6660261564999E+04	19f
16	GROW15	301	645	5665	-1.0687094129358E+08	9f
17	SCSD6	148	1350	5666	5.0500000078262E+01	24f
18	SCFXM3	991	1371	7846	5.4901254549751E+04	20f
19	SCTAP2	1091	1880	8124	1.7248071428571E+03	27f
20	GROW22	441	946	8318	-1.6083433648256E+08	10f
21	SCTAP3	1481	2480	10734	1.4240000000000E+03	28f
22	SCSD8	398	2750	11334	9.0499999992540E+02	25f

Table 4.3: Test problem dimensions.

problems was created from an example in [Chv83] and documented in [EntSS]. It was originally intended for debugging purposes. The optimal objective values for the problems as reported by Gay (excluding the DIET series) are included in the table.

Problem Number	Prob Name / # Subs	# of Steps	Min Rows	Max Rows	Min Cols	Max Cols	Min Couple	Max Couple
1	DIET2/2	2	2	3	6	6	1	1
2	DIET3/3	3	2	3	6	6	1	1
3	DIET7/7	7	2	3	6	6	1	1
4	[illegible]2205/1	18	11	15	11	14	5	5
5	SCAGR7/7	7	9	26	13	27	6	7
6	SCORPION/6	6	34	93	51	66	27	49
7	SCAGR25/25	25	19	26	13	27	7	7
8	SCTAP1/10	10	30	30	48	48	18	18
9	SCFXM1/4	4	66	92	99	126	5	9
10	GROW7/7	7	20	20	43	43	20	20
11	SCSD1/3	3	20	37	190	380	10	10
12	STAIR/6	6	38	103	71	96	46	51
13	SCRS8/7	14	50	67	350	440	10	10
14	PILOT4	4	61	154	248	252	133	154
15	SCFXM2/8	8	66	92	99	126	5	9
16	GROW15/15	15	20	20	43	43	20	20
17	SCSD6/6	7	20	20	190	210	10	10
18	SCFXM3/12	12	66	92	99	126	5	9
19	SCTAP2/10	10	109	109	188	188	62	62
20	GROW22/22	22	20	20	43	43	20	20
21	SCTAP3/10	10	148	148	248	248	86	86
22	SCSD8/7	39	10	17	70	90	10	10

Table 4.4: Test problem step dimensions.

Table 4.4 contains the staircase dimensions for each of the test problems. The default number of subproblems created is listed along with the number of steps in the staircase. The minimum and maximum dimensions of each step are given, along with the coupling between adjacent steps as described earlier in Figure 4.1.

4.2.3 Test Designs and Results

The physical properties *power*, *work* and *time* are excellent terms to describe the performance of a parallel algorithm. In the computing environment, the unit of power is a CPU, the unit of work is a CPU second, and the unit of time a second as measured with a wall clock. One can view work, or CPU time, as the rent paid

for use of the computer. The absolute performance measure, however, is usually the elapsed time needed to obtain a solution.

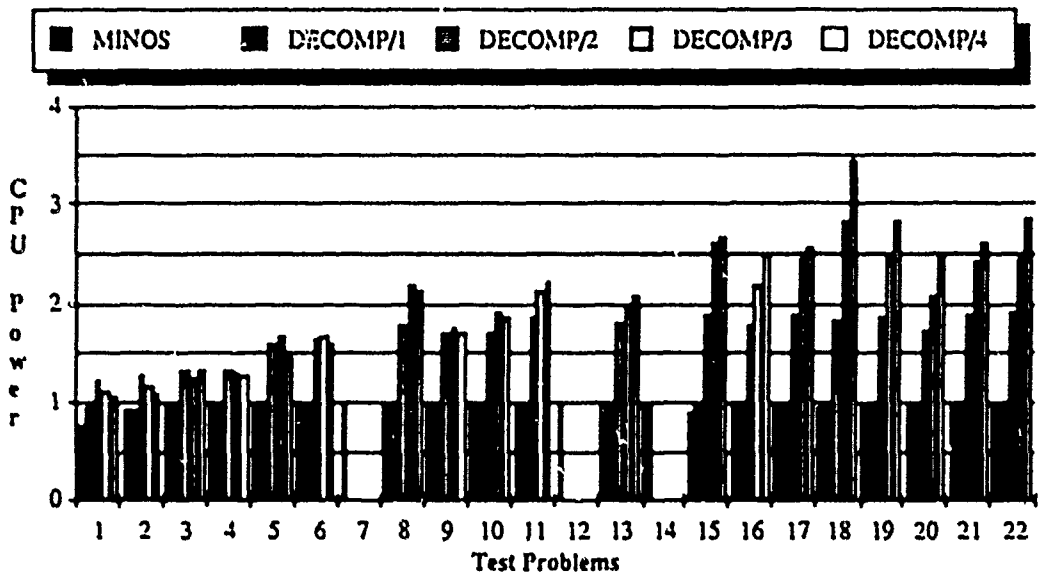

Figure 4.3: Used CPU power for each test problem.

Power: A good parallel algorithm has two properties. First, it makes efficient use of the CPU power. When two CPUs are made available, both are actually used. Most algorithms do not achieve perfect efficiency. Eighty percent is often considered very good. Figure 4.3 displays the average CPU power applied to solve each test problem when decomposed into the default number of subproblems given in Table 4.4. It shows that algorithm has little trouble utilizing more CPU power, especially on the larger problems. Of course there is a limit. Remember that at most $N-1$ processors can be kept busy by the algorithm, where N is the number of nodes/subproblems. These experiments were run with at most four CPUs because although the 3090/600E has six, it cannot effectively offer more than four CPUs in a multi-user environment.

There are no decomposition results for problems SCAGR25, STAIR and PILOT4 because dual-degeneracy prevented progress and a primal feasible solution was not obtained.

Notice also that the used CPU power for problem SC205 is at or near one regardless of p, because in this case the computer spends most of its time obtaining a primal feasible solution. At the time of writing, the Phase 1 algorithm is serial, and only one CPU is used despite the availability of more. In fact SC205 has a vacuous objective row for all but the first step in the staircase. As soon as a feasible point is found, it is the optimal one. The Phase 1 algorithm could be made parallel by passing infeasible primal solutions to the right, but this has not yet been done.

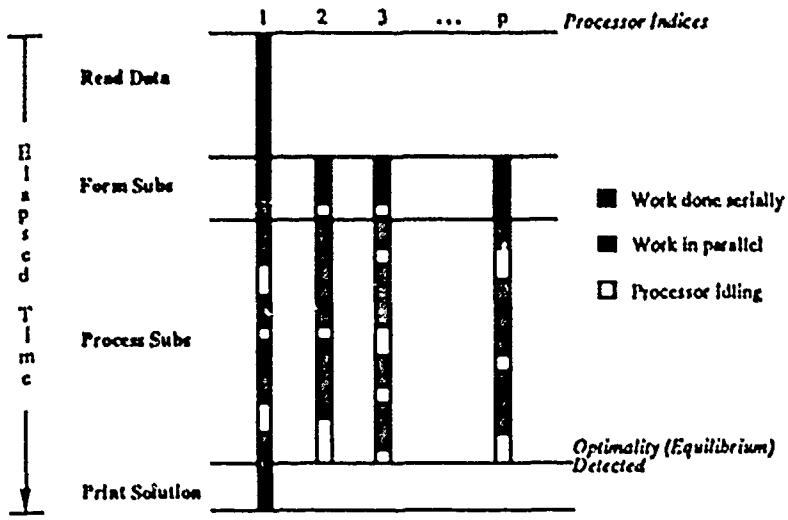

Figure 4.4: Strings of work.

Useful Work and Idle Time: Figure 4.4 will help us understand how the data for Figure 4.3 and all subsequent figures were collected. The solid lines in the figure represent useful serial work doing input and output. These times are ignored. The only times reported are those for the parallel phase, which represents a majority of the work done, especially for large problems.

After data is read from disk, the work fans out to p independent strings of work with one barrier between the Form Subs and Process Subs steps. The parallel lines in Figure 3.1 are shaded grey with intermittent white sections. This is to represent useful and idle work time. Useful work is spent forming and solving subproblems, whereas

idle work is spent counting. In the multi-user environment on the IBM 3090/600E, it is important to "waste time" counting because we must know how much idle time is really being used. It must be measured somehow. A production code would not do this. Idle time would be filled with useful work from other user's jobs. Counting idle time degrades performance at the expense of simulating a "generic" computing environment.

The CPU power in Figure 4.3 is the ratio of useful work (total length of all the grey lines) to the total work (total length of the grey and white lines). It is a measure of the effective CPU power applied to solving the problem.

A second aspect of collecting CPU times needs to be reported. Each parallel string of work is implemented as a series of MVS operating system tasks, the number of which is not predetermined. Partly because of this, the IBM Parallel Fortran Compiler has no facility for collecting individual CPU times. An assembler language routine for collecting MVS task times was used instead.

On every call to the Parallel Fortran Library the MVS task may change. This has been likened to taking a sequence of taxis to travel to some destination. Street intersections represent library calls. You never know when you will change taxis, so to ensure payment, you make installments for each block driven. The time spent crossing intersections is not recorded. Likewise, the MVS task time is recorded between subroutine calls, but the time spent *in* the subroutine library is not recorded, and causes a 10% to 15% shortfall in the total CPU time reported for the largest test problem SCSD8; see Table 4.5. The unaccounted time falls into the idle-work category because the library routines are called only when a processor is trying to find something to do besides count. For this reason, the clocks on each Fortran Processor are used to measure only useful work, while a job clock measures the total length of all the parallel strings. The difference between the sum of the processor clocks and the job clock is attributed to idle work.

Problem Name	Real CPU	My CPU	Percent Error
SCSD8/2	15.32	14.106	8%
SCSD8/3	12.74	11.569	9%
SCSD8/4	12.78	11.587	9%
SCSD8/5	10.91	9.693	11%
SCSD8/6	12.13	10.908	10%
SCSD8/7	10.72	9.471	12%
SCSD8/8	13.58	12.312	9%
SCSD8/9	9.73	8.488	13%
SCSD8/10	10.97	9.708	12%
SCSD8/11	11.59	10.288	11%
SCSD8/12	11.42	10.072	12%
SCSD8/13	10.89	9.575	12%
SCSD8/14	10.71	9.377	12%
SCSD8/39	31.55	28.967	8%

Table 4.5: Shortfalls in measuring work.

Another set of runs were executed with and without the processor clocks, providing a very sensible illustration of the Heisenberg Principle. *You cannot measure performance without affecting it.*

Job CPU times are reported in Table 4.5. SCSD8 was solved multiple times for $n = 11$ and $p = 1, 4$ both with and without the processor clocks. Two percent faster times were obtained without the clocks when using only one processor, and four percent *slower* times were obtained without the clocks when using four processors. We can expect a similar effect for other test problems.

Two speculations have been offered to support the variations caused by alling the clocks. The first is that part of the excess time is getting lost by the operating system during the system clock calls [Wel89]. The other is that the operating system is using the clock calls as opportunities to interrupt the processor [For89]. A system interrupt would appear beneficial in that it would most likely be interrupting idle work time.

Finally, we can see from Figure 4.5 that the total work required to solve a problem is not deterministic. The same program configuration was run several times with

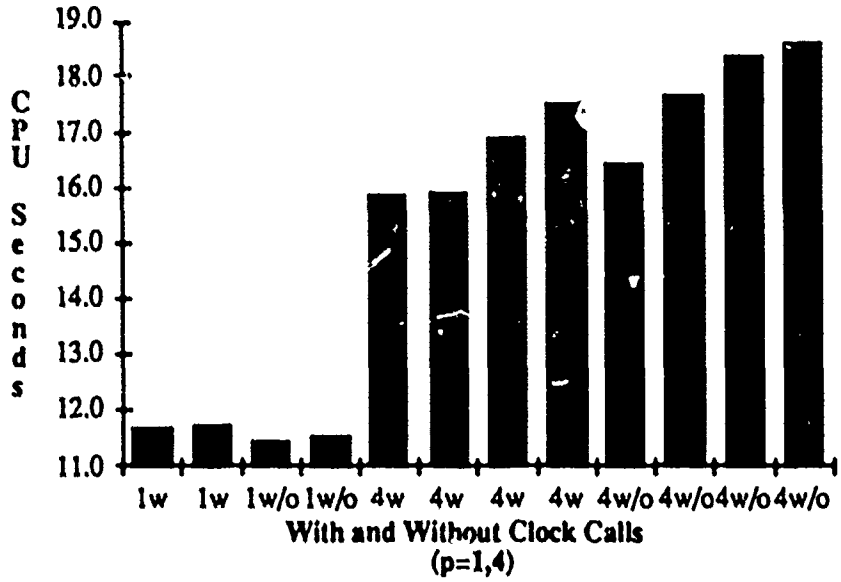

Figure 4.5: The Heisenberg Principle.

different results. Hence, *the reported times are the average of up to three successive runs.*

Work: The second property of a good parallel algorithm is that the total work does not increase as the number of processors increases. Figure 4.6 gives the total parallel work done on each test problem using both MINOS and DECOMP ($p = 1, 2, 3, 4$). Notice that the work actually decreases from $p = 1$ to $p = 2$ for problem 13. This is possible because there is no control over the path taken to the solution, and different paths can be taken for different numbers of processors. The conclusion to be drawn from these results is that the total work does not substantially increase as the number of processors increases.

Time: Together with the effective use of CPU power, we obtain respectable reductions in elapsed times to solve the test problems, as shown in Figure 4.7. We have used a log scale for this figure because of the great disparity in time required to solve

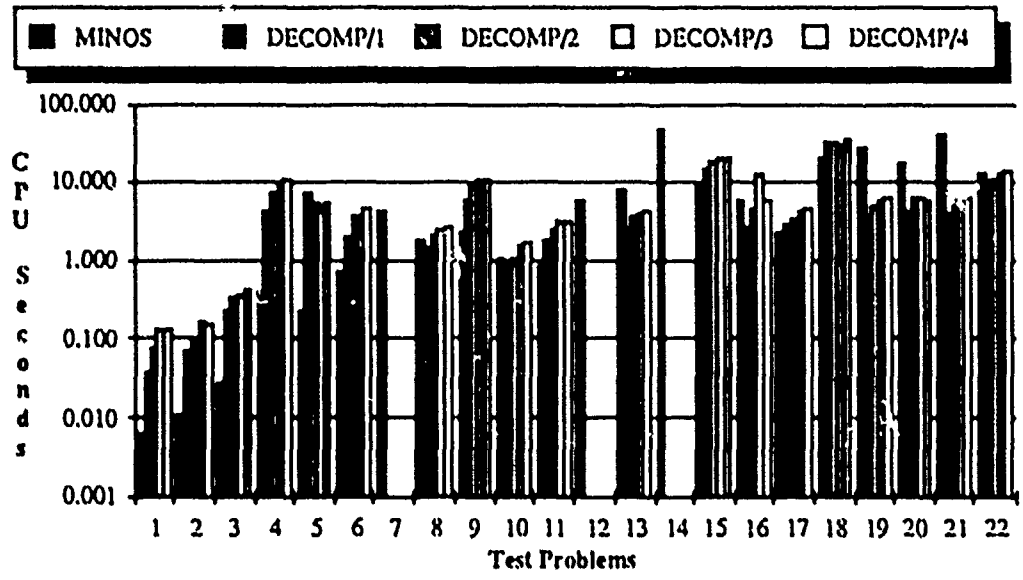

Figure 4.6: Work required to solve each test problem.

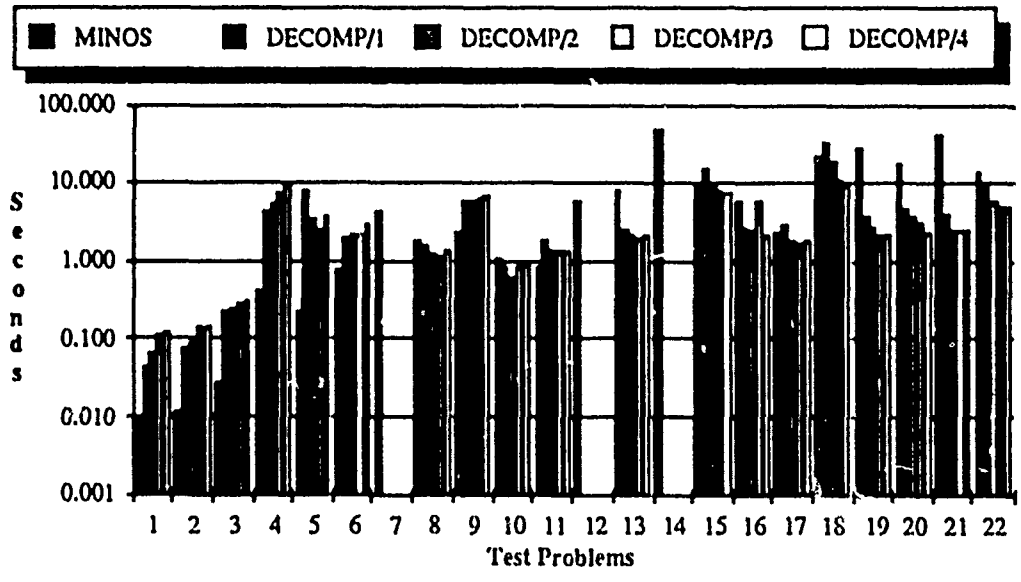

Figure 4.7: Time required to solve each test problem.

the small and the large problems. A more effective presentation is made by normalizing the scale for each tes problem. In Figure 4.8, the times of each individual test problem have been normalized by the time used by MINOS. The result is called the "Speedup over MINOS." In this case, a value of 2 would mean that the decomposition algorithm found the solution twice as fast as MINOS. The figure shows that parallel decomposition is consistently better than the simplex method on the larger problems.

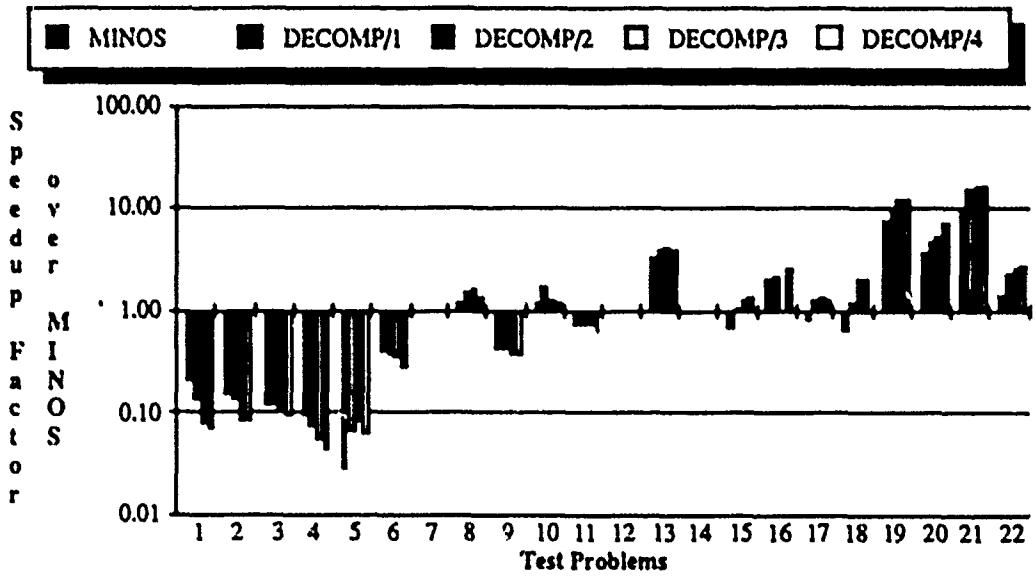

Figure 4.8: Speedup over MINOS for each test problem.

Speedups: There are two benefits derived from parallel decomposition that give such speedups. The first is that for the larger problems, decomposition alone ($p = 1$) has offered a speedup. For instance, problem 21 (SCTAP3) is solved 10.5 times faster just because of a change of algorithm.

The second benefit, naturally, is derived from using more CPU power. Figure 4.9 is a display of elapsed times that were normalized by the time used by DECOMP/1 for each test problem. With this perspective, we can effectively judge the benefits of adding processors. Notice that for problem 21, the computation was sped up by an *additional factor* of 1.6 over DECOMP/1 because of the addition of three

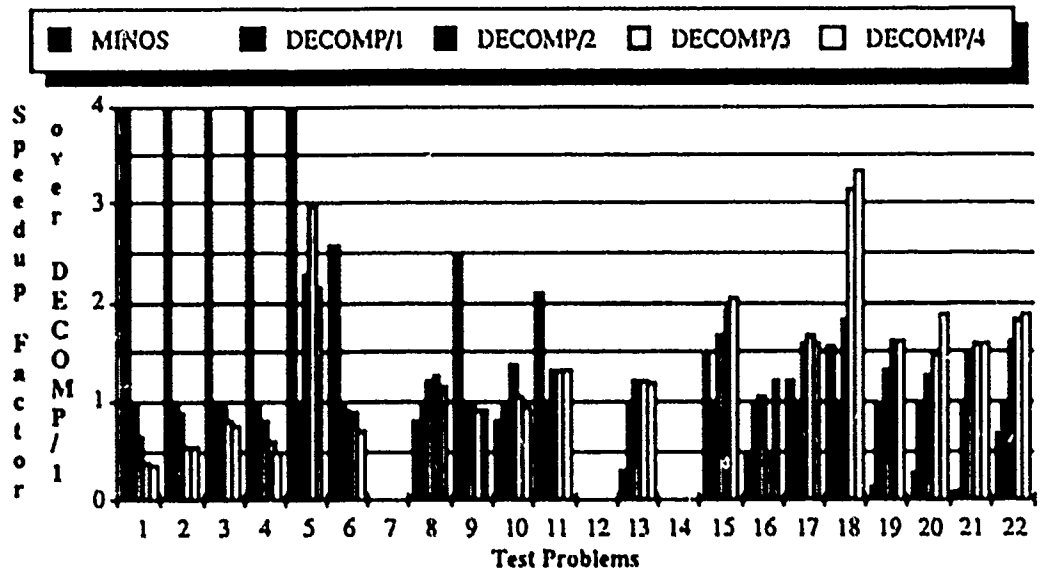

Figure 4.9: Speedup over DECOMP/1 for each test problem.

CPUs of power for a total for four CPUs. The overall benefit provided by parallel decomposition with four processors was a factor of 16.8 speedup as seen in Figure 4.8.

The Number of Subproblems: This experiment demonstrates the increase in overhead of decomposition as the number of subproblems increases. The largest test problem, SCSD8, has 39 steps. If subproblems are limited to a discrete number of steps, the number of subproblems is limited to the set $\{2, 3, \ldots, 39\}$. Note also that there are 38 ways to partition the two-subproblem case. The number of steps per subproblem was chosen to be nearly the same in each case.

Even though communication times are negligible, because this is a shared memory computer, there is still a significant overhead involved in making the proper response to all received messages. On the other hand, there is an uncertain benefit from solving a staircase with decomposition. These two effects combine in this experiment. Figure 4.10 shows the total work used to solve SCSD8 when the number of subproblems N, varies between 2 and 38 (even numbers only), and the number of processors p, varies from one to four.

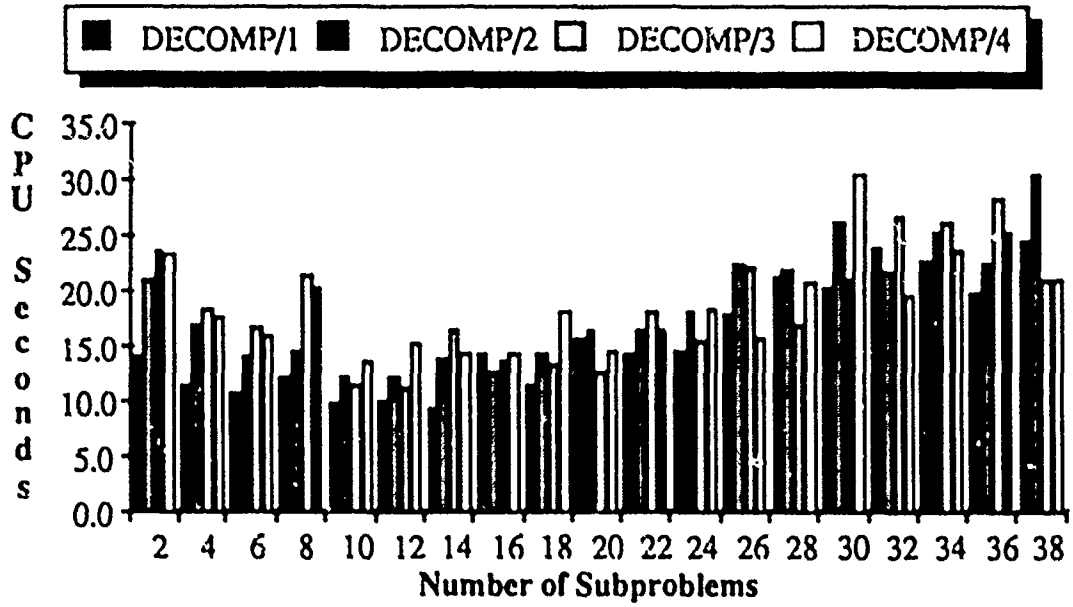

Figure 4.10: Work to solve SCSD8 using increasingly finer partitions.

For small N, the total work remains relatively constant—around ten CPU seconds. The run requiring the least amount of work is for $N = 14$ and $p = 1$ with 9.4 CPU seconds. After the average size of a subproblem begins to fall below three steps, $N \approx 13$, the total work increases. This is consistent with our general observations with one CPU, that a staircase problem with less than 2000 nonzeros is not worth decomposing, as the decomposition overhead begins to outweigh its benefits. However, with more processors the results are different.

The run solving the fastest overall, as seen in Figure 4.11, is $N = 10$, $p = 4$ with 3.4 elapsed seconds. For $p = 2$, the minimum is 5.8 elapsed seconds with $N = 10$ and 12, and for $p = 3$ the minimum is 3.7 elapsed seconds occurring at both $N = 12$.

The best speedup for a fixed number of subproblems is at $N = 32$, with a factor of 4.7; see Figure 4.12. However, the best serial time versus the best parallel time is $8.7/3.4 = 2.6$, but this was obtained only after an exhausting search.

The trend is consistent that more processors makes decomposition faster. The effect of the number of processors on solution time is likely to be a function of the

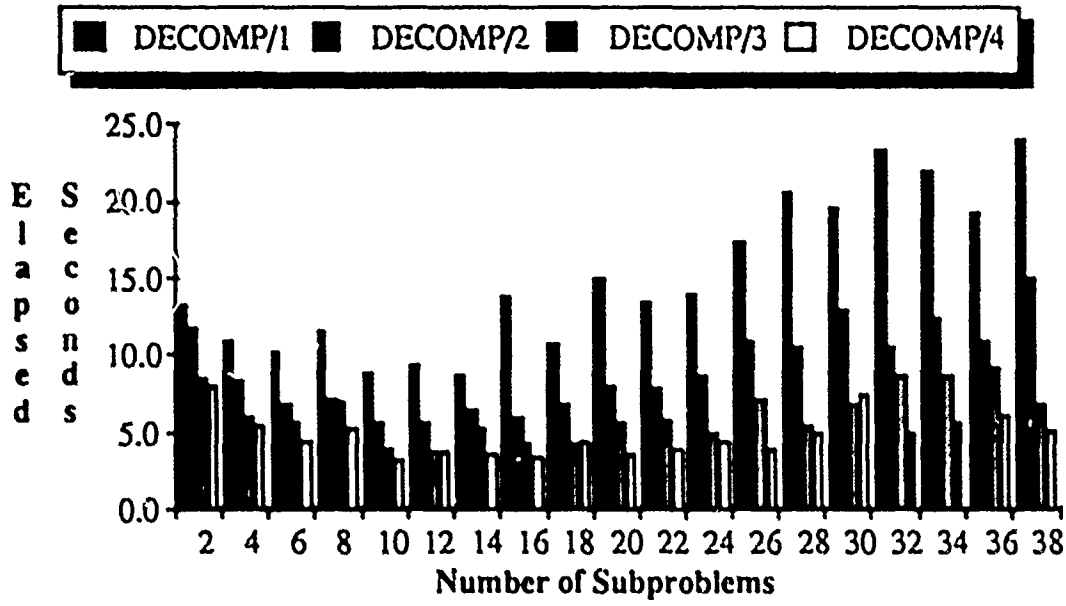

Figure 4.11: Time versus the number of subproblems for SCSDS.

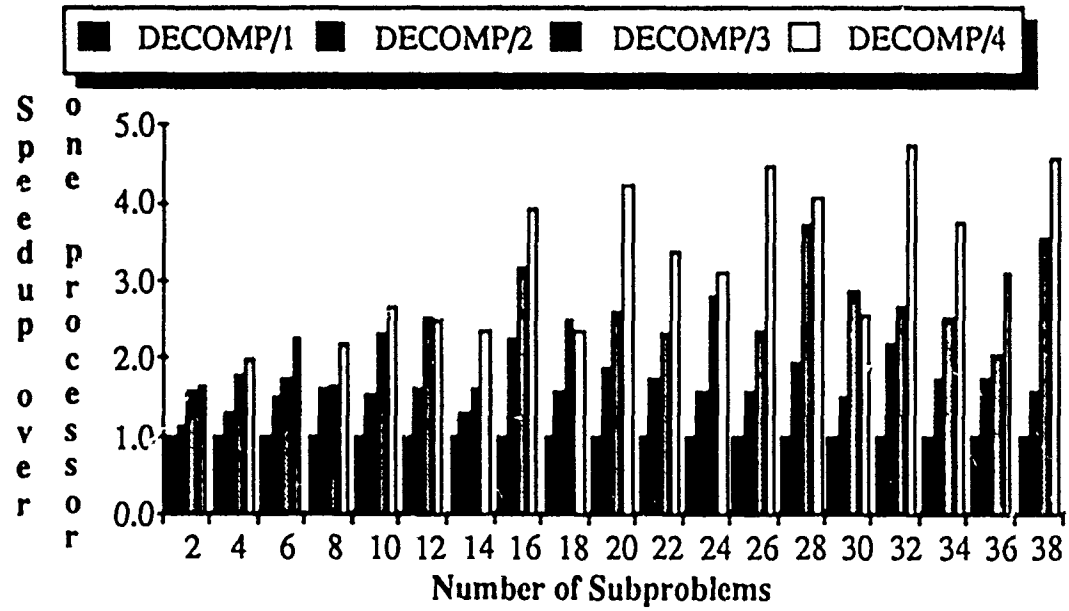

Figure 4.12: Speedup versus the number of subproblems for SCSDS.

solver more than anything else. Small LP subproblems are best solved with a vectorized tableau method. Specialized subproblems like network flows are best solved by combinatoric algorithms. MINOS, the solver used here, performs best on medium-sized staircase problems (relative to decomposition). Serial solvers should be chosen according to the size and nature of the subproblems.

The Number of Processors: This is a study on the effective use of processors. At a time when the computer was lightly loaded, SCSDS was solved using from one to seven Fortran Processors. In IBM Parallel Fortran, a Fortran Processor is a series of MVS Operating System tasks, so more than six may be requested for a six-processor machine. Seven is the limit based on memory restrictions.

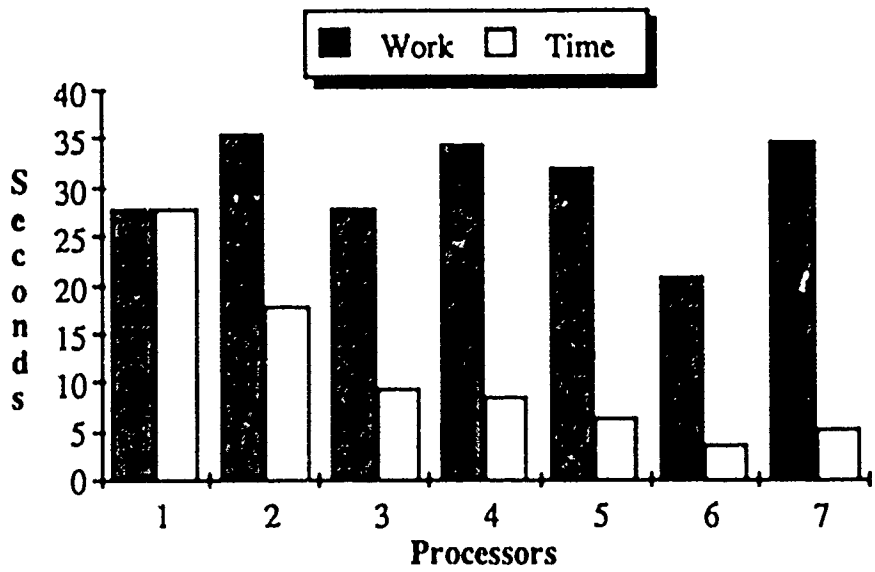

Figure 4.13: Work and time versus processors for SCSDS.

Figure 4.13 is a classic speedup diagram for this problem. Here, speedup is calculated relative to the solution time for decomposition with one processor. Naturally, the point (1,1) is represented. The diagonal line shows the ideal.

The next figure, 4.14, graphs the dichotomy of Work versus Time for varying numbers of processors. Sharp dips in the amount of work, as for the six-processor

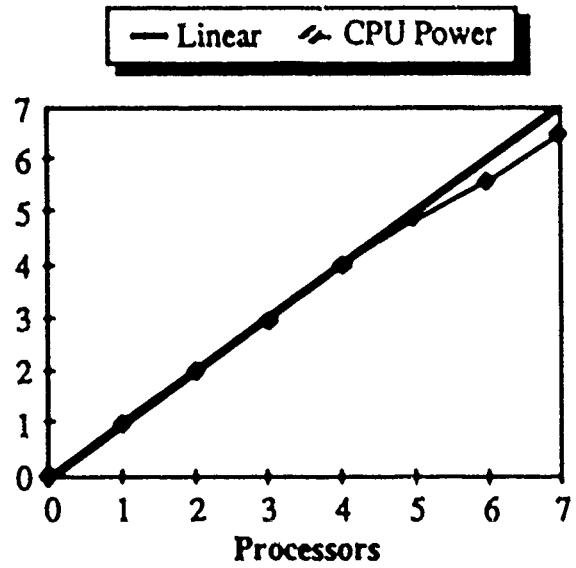

Figure 4.14: Power versus processors for SCSDS.

case, can only be attributed to good fortune. Experiments on a dedicated machine could settle many uncertainties as to the true benefactors of parallel decomposition. At this writing, we can say only that they exist.

4.2.4 Performance Extrapolations

How will parallel decomposition perform on larger problems?

In the next two experiments, we reexamine the results for a constant number of subproblems by grouping the test problems by family. We consider, as the problems in a family get larger, how parallel decomposition should perform on even larger problems.

Extending the Staircase: This is the first of two discussions regarding extrapolation of the results beyond the test suite. One way to make a staircase problem larger is to add more steps. This means that either the planning horizon is lengthened or it is represented in finer detail. There are three such series in our test suite:

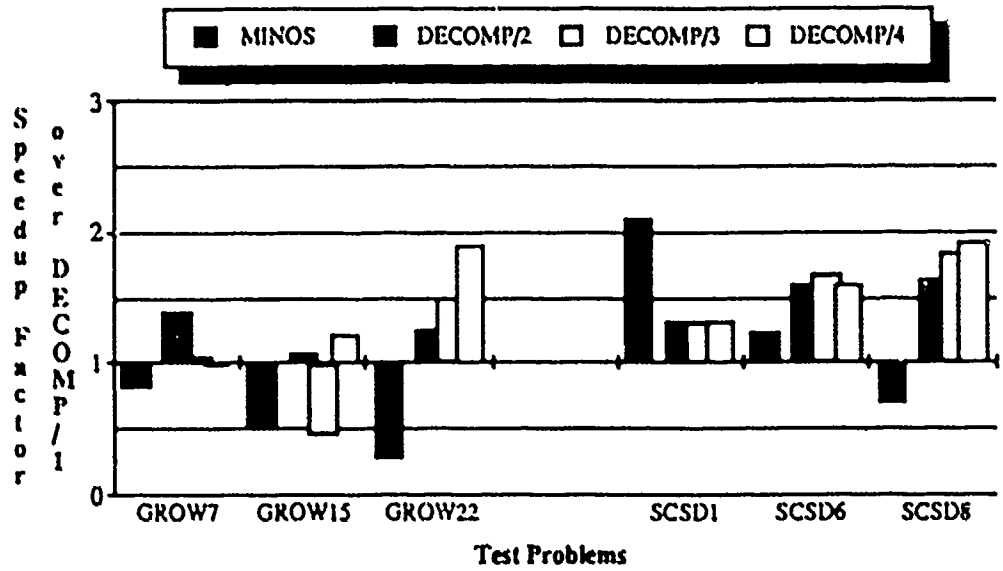

Figure 4.15: Speedup over DECOMP/1 for extending staircases.

DIET, GROW, and SCSD. The speedup results from Figure 4.9 are reproduced in Figure 4.15 for the latter two only, since the LPs of the DIET series are too small. We see that as the length of the staircase extends, the parallel algorithm's performance is not degraded.

Model Complexity: Another method of increasing the size of staircase problems is to add more complexity to the model,i.e., to disaggregate. For instance, "dairy products" becomes milk, cheese, yogurt and ice cream. Adding complexity allows a model to give a more detailed solution, and the modeler to address interactions more specifically. A summer rise in the price of the aggregate "dairy products" may only be a reflection of more demand for ice cream!

The SCTAP series of problems keep the same number of steps, but increase the number of rows, columns and nonzeros per step. Figure 4.16 is a reproduction of the elapsed times for this series. It shows that the simplex method has increasing difficulty with this problem, while the performance of the parallel decomposition algorithm does not degrade.

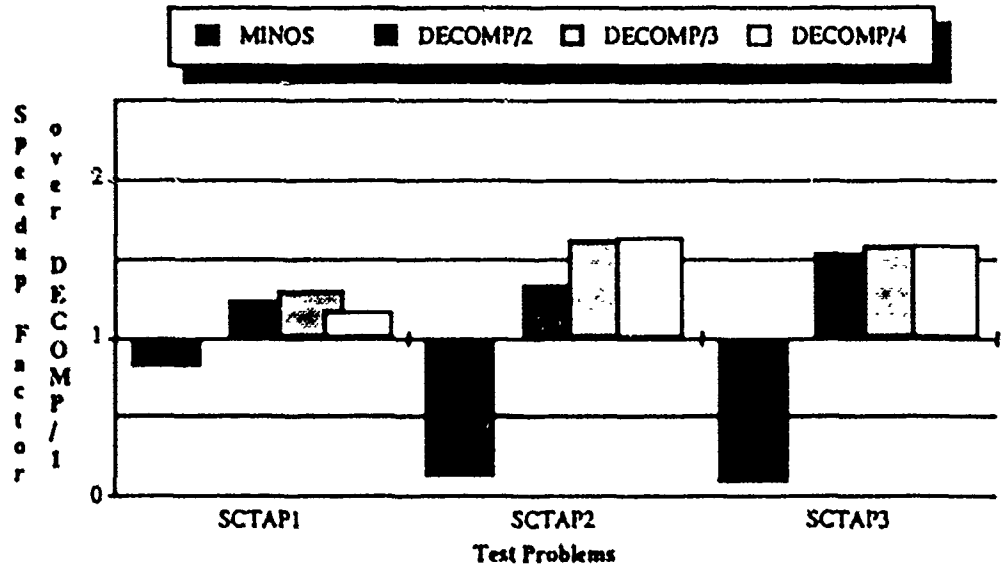

Figure 4.16: Speedup over DECOMP/1 for more complex staircases.

4.3 Conclusions

We have taken a long tour through the space of all communication networks, but the experience has created surgeons from interns. What we slice apart is more than just a linear program. It is a modeler's presentation of some small part of the world. The pieces and their interactions can now be observed from a new perspective: as a network of communicating entities. The communication is structured and directed toward obtaining a consensus via local agreements. How can communication patterns be studied? Are their optimal configurations based on a modeler's knowledge of the natural configuration? What are the strong and the weak links? These are probing questions to answer with further investigation.

The main conclusion to make about the computational results is that if serial decomposition does well on a given problem then parallel decomposition does also. This is not surprising, but what we have also seen is that even when serial decomposition is slow, parallel decomposition can still be made to solve proolems faster than the

simplex method by adding more processors. In general, adding more processors will help, but there is a limit.

An important accomplishment is that by characterizing the oracle and the relaxed oracle, we define an interface that allows any convenient subproblem solver to be used. The essential part of decomposition is not *how* a subproblem is solved, but the *form* of its solution.

In addition, we can now see that the subproblems need not be linear programs. Convex functions and regions can be approximated with piece-wise linear functions and extreme-point representations.

Finally, no practical implementation of a theoretical algorithm is perfect. Ours needs work to make it more robust and handle ever larger problems. Let it be our hope that the techniques and ideas discussed here will find practical use.

Appendix A

Example Subproblem Formulations

We now present examples of decomposition applied to three structured linear programs, and one that is unstructured. These are intended to offer a better understanding of the previous sections, and serve as recommended procedures for applying the concepts of this thesis to practical examples.

Block Diagonal: This is the simplest example for decomposition. The problem consists of two completely independent linear programs contained in one. By investigating the formulations of the subproblems, we find that decomposition can impose dependencies not regularly recognized in practice.

Staircase: Here we take a staircase pattern and slice it vertically just as in the diagram in the introduction of this thesis. In the final chapter, we apply the parallel oracle to the resulting subproblems for a variety of real-world test problems.

Two-Stage Stochastic: This is our first example of cross nesting. Again, the diagram in the introduction contains the anatomy and the sequence of slices used.

Dense: This nondescript structure is used to demonstrate a procedure by which the anatomic structure is broken down to the level of a single coefficient.

These examle don not make use of hte subproblem interface theorem, so for instance a vertical arc index set is defined on the intersection of the row index sets of the joined nodes.

A.1 Block Diagonal Example

We begin our series of examples with the simplest block diagonal case, where the constraints of two subproblems lie in independent spaces. The subproblems are completely independent, except that they are coupled via the objective, indexed by σ. In this example there will be information passed between the subproblems, but only of the most trivial nature.

$$\begin{array}{lll} \min\limits_{\substack{x^1\geq 0\\ x^2\geq 0}} & & c^1x^1 + c^2x^2 = z \\ \text{s.t.} & \pi^1: & A^{11}x^1 \geq b^1 \\ & \pi^2: & A^{22}x^2 \geq b^2. \end{array} \tag{A.1}$$

As noted earlier, the names of the primal and dual variables of the Block Diagonal Problem are used as indices for the rows and columns of the coefficient matrix A. The block diagonal LP (A.1) has superscripts in order to differentiate the problem data and variables from those of the subproblems, which will have subscripts.

Block Diagonal Problem Description:

$$\mathcal{R} = \pi^1 \cup \pi^2, \quad \mathcal{C} = x^1 \cup x^2,$$

$$A = \begin{pmatrix} A^{11} & 0 \\ 0 & A^{22} \end{pmatrix} \in \Re^{\mathcal{R}\times\mathcal{C}}, \quad b = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix} \in \Re^{\mathcal{R}}, \quad c^T = \begin{pmatrix} c^1 & c^2 \end{pmatrix} \in \Re^{1\times\mathcal{C}}.$$

Block Diagonal Communication Network Description:

$$\begin{aligned} &(\mathcal{N}, \mathcal{A}) = (\{1,2\}, \{(12),(21)\}) \\ &\mathcal{R}_1 = \pi^1, \quad \mathcal{R}_2 = \pi^2, \\ &\mathcal{C}_1 = x^1 \cup x^2, \quad \mathcal{C}_2 = x^1 \cup x^2, \\ &\mathcal{T}_{12} = \text{down}, \quad \mathcal{T}_{21} = \text{up}. \end{aligned}$$

Block Diagonal Incidence Graph Description:

$$h = (\{\sigma, \pi^1, \pi^2, s, x^1, x^2\}, \{(\sigma x^1), (\sigma x^2), (\pi^1 s), (\pi^1 x^1), (\pi^2 s), (\pi^2 x^2)\}).$$

Recall that σ and s index the objective and right-hand side, respectively.

Block Diagonal Arc Index Sets: There are no horizontal arcs, so $\mathcal{C}_{12} = \mathcal{C}_{21} = x^2$.

Block Diagonal Partition Graphs: There is only one partition graph, so all added variables will be indexed by their associated arcs. $p = (\mathcal{N}, \mathcal{A})$, $\mathcal{R}_p = \mathcal{R}$, $\mathcal{C}_p = \mathcal{C}$.

Block Diagonal Subproblem ($1 \in \mathcal{N}$): Node one is topmost and leftmost.

Original Variables: $\pi_1 \in \Re^{\mathcal{R}_1}$ and $x_1 \in \Re^{\mathcal{C}_1}$.

Original Data: $A_1 = (A^{11} \quad 0)$, $b_1 = (b^1)$, and $c_1^T = (c^1 \quad c^2)$.

Incoming Arcs: There is one incoming arc to node one, (21), and it has type $\mathcal{T}_{21} =$ up. It determines the added variables and data.

Added Variables: $l_{21} \in \Re^{K_2}$, $\theta_{21} \in \Re$, and $\psi_{21} \in \Re^{\mathcal{C}_{21}}$.

Added Data: 1 is indexed by (θ_{21}, s_1), $\tilde{g}_{12}$ is indexed by $(\mathcal{C}_{21}, l_{21})$, $\tilde{X}_{12}$ is indexed by (ψ_{21}, l_{21}), and $-I_{12}$ is indexed by (ψ_{21}, x_1).

Non-negativity: $l_{21} \geq 0$ and x_1 is free.

Constraint Types: π_1 is a $\geq$, ψ_{21} is an $=$, and θ_{21} is an $=$.

Formulation ($1 \in \mathcal{N}$):

	l_{21}	x_1		s_1
π_1		A_1	$\geq$	b_1
ψ_{21}	$\tilde{X}_{12}$	$-I_{12}$	$=$	0
θ_{21}	$\tilde{g}_{12}^T$		$=$	1
	≥ 0	*free*		
σ_1	0	c_1^T		

Notice in the formulation for node 1 that for the dual variables, $\psi_{21} = c^1$ because of a dual identity with the objective row. This means that the information being passed to node 2 is constant and equals the values of the original objective for the columns x^1.

Block Diagonal Subproblem ($2 \in \mathcal{N}$): Node two is leftmost and not topmost.

Original Variables: $\pi_2 \in \Re^{\mathcal{R}_2}$ and $x_2 \in \Re^{\mathcal{C}_2}$.

Original Data: $A_2 = (0 \quad A^{22})$, $b_2 = (b^2)$, $c_2^T = (0 \quad 0)$.

Incoming Arcs: There is one incoming arc (12), of type $\mathcal{T}_{12} =$ down.

Added Variables: $w_{12} \in \Re$ and $v_{12} \in \Re$.

Added Data: $\tilde{\psi}_{21}$ is indexed by (v_{12}, x_2), $\tilde{\theta}_{21}$ is indexed by (v_{12}, s), $\tilde{\delta}_{21}$ is indexed by (v_{12}, w_{12}), and another $\tilde{\delta}_{21}$ is indexed by (σ_2, w_{12}).

Non-negativity: $x_2 \geq 0$, and w_{12} is free.

Constraint Types: v_{12} is $\geq$ and π_2 is $\geq$.

Formulation ($2 \in \mathcal{N}$):

	x_2	w_{12}		s_2
v_{12}	$\tilde{\psi}_{21}^T$	$\tilde{\delta}_{21}$	$\geq$	$-\tilde{\theta}_{21}$
π_2	A_2		$\geq$	b_2
	≥ 0	*free*		
σ_2	0	$\tilde{\delta}_{21}$		

The node 2 subproblem will be solved only once based on the constant information passed to it from node 1. The returned primal solution, when incorporated into $\tilde{X}_{12}$ and $\tilde{g}_{12}$ in the node-1 subproblem, will allow it to be solved in only one iteration. The overall optimum is then achieved. The overall solution is $(\begin{pmatrix} \tilde{\pi}_1 \\ \tilde{\pi}_2 \end{pmatrix}, \tilde{x}_1)$.

A.2 Staircase Example

This example differs from the previous in that it uses Benders Decomposition and there are now coupling constraints between the partitioned columns. As a result, the information passed over the communication network will not be so trivial.

$$\begin{array}{lll} \min\limits_{\substack{x^1 \geq 0 \\ x^2 \geq 0}} & c^1x^1 + c^2x^2 = z & \\ \text{s.t.} \quad \pi^1: & A^{11}x^1 \geq b^1 & \\ \pi^2: & A^{21}x^1 + A^{22}x^2 \geq b^2. & \end{array} \tag{A.2}$$

As in the previous example, the names of the primal and dual variables of the Staircase Problem are used as indices for the rows and columns of the matrix A.

Staircase Problem Description:

$$\mathcal{R} = \pi^1 \cup \pi^2, \quad \mathcal{C} = x^1 \cup x^2,$$

$$A = \begin{pmatrix} A^{11} & 0 \\ A^{21} & A^{22} \end{pmatrix} \in \Re^{\mathcal{R} \times \mathcal{C}}, \quad b = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix} \in \Re^{\mathcal{R}}, \quad c^T = \begin{pmatrix} c^1 & c^2 \end{pmatrix} \in \Re^{1 \times \mathcal{C}}.$$

Staircase Communication Network Description:

$$(\mathcal{N}, \mathcal{A}) = (\{1,2\}, \{(12),(21)\})$$
$$\mathcal{R}_1 = \pi^1 \cup \pi^2, \quad \mathcal{R}_2 = \pi^1 \cup \pi^2,$$
$$\mathcal{C}_1 = x^1, \quad \mathcal{C}_2 = x^2,$$
$$\mathcal{T}_{12} = \text{right}, \quad \mathcal{T}_{21} = \text{left}.$$

Staircase Incidence Graph Description:

$$h = (\{\sigma, \pi^1, \pi^2, s, x^1, x^2\}, \{(\sigma x^1),(\sigma x^2),(\pi^1 s),(\pi^1 x^1),(\pi^2 s),(\pi^2 x^1),(\pi^2 x^2)\}).$$

Staircase Arc Index Sets: There are no column coupling sets since there are no vertical arcs. $\mathcal{R}_{12} = \mathcal{R}_{21} = \pi^2$.

Staircase Partition Graphs: There is only one partition graph, so all added variables will be indexed by their associated arcs. $p = (\mathcal{N}, \mathcal{A})$, $\mathcal{R}_p = \mathcal{R}$, and $\mathcal{C}_p = \mathcal{C}$.

Staircase Subproblem ($1 \in \mathcal{N}$): Node one is topmost and leftmost.

Original Variables: $\pi_1 \in \Re^{\mathcal{R}_1}$, and $x_1 \; in \Re^{\mathcal{C}_1}$.

Original Data: $A_1 = \begin{pmatrix} A^{11} \\ A^{21} \end{pmatrix}$, $b_1 = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix}$, and $c_1^T = (c^1)$.

Incoming Arcs: There is one incoming arc to node one and its type is $\mathcal{T}_{21} = \text{left}$.

Added Variables: $\lambda_{21} \in \Re^{K_2}$, $t_{21} \in \Re$, and $y_{21} \in \Re^{\mathcal{R}_{21}}$.

Added Data: 1 is indexed by (σ_1, t_{21}), $\tilde{\gamma}_{12}$ is indexed by (λ_{21}, t_{21}), $\tilde{\Pi}_{12}$ is indexed by (λ_{21}, y_{21}), and $-I_{12}$ is indexed by (π_1, y_{21}).

Non-negativity: $x_1 \geq 0$, and both y_{21} and t_{21} are free.

Constraint Types: λ_{21} is $\geq$ and π_1 is $=$.

Formulation ($1 \in \mathcal{N}$):

	x_1	$\tilde{y}_{21}$	t_{21}		s_1
λ_{21}		$\tilde{\Pi}_{12}$	$\tilde{\gamma}_{12}$	$\geq$	0
π_1	A_1	$-I_{12}$		$=$	b_1
	≥ 0	*free*	*free*		
σ_1	c_1^T	0	1		

Staircase Subproblem ($2 \in \mathcal{N}$): Node two is topmost and not leftmost.

Original Variables: $\pi_2 \in \Re^{\mathcal{R}_2}$ and $x_2 \in \Re^{\mathcal{C}_2}$.

Original Data: $A_2 = \begin{pmatrix} 0 \\ A^{22} \end{pmatrix}$, $b_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and $c_2^T = (c^2)$.

Incoming Arcs: There is one arc (12) incident to node one and its type is $\mathcal{T}_{12} =$ right.

Added Variables: $\omega_{12} \in \Re$, and $u_{12} \in \Re$.

Added Data: $\tilde{y}_{21}$ is indexed by (π_2, u_{12}), $-\tilde{t}_{21}$ is indexed by (σ_2, u_{12}), $\tilde{d}_{21}$ is indexed by (ω_{12}, u_{12}), and another $\tilde{d}_{21}$ is indexed by (ω_{12}, s_2).

Non-negativity: $u_{12} \geq 0$ and $x_2 \geq 0$.

Constraint Types: π_2 is $\geq$, ω_{12} is $= 1$.

Formulation ($2 \in \mathcal{N}$):

	u_{12}	x_2		s_2
π_2	$\tilde{y}_{21}$	A_2	$\geq$	0
ω_{12}	$\tilde{d}_{21}$		$=$	$\tilde{d}_{21}$
	≥ 0	≥ 0		
σ_2	$-\tilde{t}_{21}^T$	c_2^T		

A.3 Two-Stage Stochastic Example

Consider b^2 in (A.3) to be a discrete random variable having realizations b^2_s and probabilities P_s for all $s \in \{1, \ldots S\}$. We will derive the subproblems for this multi-point distribution.

$$\begin{array}{lll} \min\limits_{\substack{x^1 \geq 0 \\ x^2 \geq 0}} & & c^1x^1 + \ c^2x^2 = z \\ \text{s.t.} & \pi^1 : & A^{12}x^2 \geq b^1, \\ & \pi^2 : & A^{21}x^1 + A^{22}x^2 \geq b^2. \end{array} \tag{A.3}$$

Stochastic Problem Description: For a multi-point distribution, the indices π^2_s and x^2_s will be repeated for each instance of s. However, it is a modeling issue as to whether π^1 is repeated. The meaning of these constraints becomes ambiguous when random data are introduced. If we limit ourselves to linear formulations, we still have the choice to model them either as a single expected-value constraint, $E_s\{A^{21}x^2_s\} \geq b^1$, or as multiple absolute constraints $A^{21}x^2_s \geq b^1$, for all s.

Given the *a priori* assumption to keep the objective linear by using expected values, the second case corresponds to Stochastic Linear Recourse. To keep things simple we choose the expected-value constraint.

$$\mathcal{R} = \pi^1 \cup \pi^2_s, \quad \mathcal{C} = x^1 \cup x^2_s,$$

$$A = \begin{pmatrix} 0 & A^{12}_1 & \cdots & A^{12}_S \\ A^{21} & A^{22} & & \\ \vdots & & \ddots & \\ A^{21} & & & A^{22} \end{pmatrix} \in \Re^{\mathcal{R}\times\mathcal{C}}, \quad b = \begin{pmatrix} b^1 \\ b^2_s \end{pmatrix} \in \Re^{\mathcal{R}}, \quad c^T = \begin{pmatrix} c^1 & c^2_s \end{pmatrix} \in \Re^{1\times\mathcal{C}},$$

$$A^{12}_s \doteq P_s A^{12}, \quad c^2_s \doteq P_s c^2.$$

Stochastic Communication Network Description: This example crosses D-W and Benders Decomposition. First D-W is applied, then Benders is applied to the bottom problem. This obviates the need for the special Cross-Splitting described

previously. The communication network is defined for all $s \in \{1,\ldots,S\}$.

$$(\mathcal{N},\mathcal{A}) = (\{1,2,s\},\{(12),(21),(1s),(s1),(2s),(s2)\}),$$
$$\mathcal{R}_1 = \pi^1, \quad \mathcal{R}_2 = \pi^2_s \forall x \in \{1,\ldots,S\}, \quad \mathcal{R}_s = \pi^2_s,$$
$$\mathcal{C}_1 = x^1 \cup x^2_s \forall x \in \{1,\ldots,S\}, \quad \mathcal{C}_2 = x^1, \quad \mathcal{C}_s = x^2_s,$$
$$\mathcal{T}_{12} = \text{down}, \quad \mathcal{T}_{21} = \text{up}, \quad \mathcal{T}_{1s} = \text{down}, \quad \mathcal{T}_{s1} = \text{up}, \quad \mathcal{T}_{2s} = \text{right}, \quad \mathcal{T}_{s2} = \text{left}.$$

In addition to nodes one and two, this network has one node for each distribution point. Each communicates to node 1 via up and down arcs, and with node 2 via left and right arcs.

Stochastic Incidence Graph Description: Note that the nodes for π^2_s and x^2_s are repeated for each instance of s.

$$h = (\{\sigma,\pi^1,\pi^2_s,s,x^1,x^2_s\},\{(\sigma x^1),(\sigma x^2_s),(\pi^1 s),(\pi^1 x^2_s),(\pi^2_s s),(\pi^2_s x^1),(\pi^2_s x^2_s)\}).$$

Stochastic Arc Index Sets: Nodes 1 and 2 communicate only objective information as we saw previously in the Block Diagonal Example.

$$\mathcal{R}_{2s} = \mathcal{R}_{s2} = \pi^2_s, \quad \mathcal{C}_{12} = \mathcal{C}_{21} = x^1, \quad \mathcal{C}_{1s} = \mathcal{C}_{s1} = x^2_s.$$

Stochastic Partition Graphs: There are two partition graphs p_1 and p_2. They are ordered so that p_1 is before (i.e., the parent of) p_2.

$$p_1 = (\{1,2,s\},\{(12),(21),(1s),(s1)\}), \quad \mathcal{R}_{p_1} = \mathcal{R}, \quad \mathcal{C}_{p_1} = \mathcal{C},$$

$$p_2 = (\{2,s\},\{(2s),(s2)\}), \quad \mathcal{R}_{p_2} = \pi^2_s, \quad \mathcal{C}_{p_2} = \mathcal{C}.$$

The added variables associated with arcs (21) and (s1), which link the child partition nodes $\{2,s\}$ to the parent partition node $\{1\}$, will be indexed by the child partition p_2.

Stochastic Subproblem ($1 \in \mathcal{N}$): Node one is topmost and leftmost.

Original Variables: $\pi_1 \in \Re^{\mathcal{R}_1}$, and $x_1 \in \Re^{\mathcal{C}_1}$.

Original Data: $A_1 = (0 \quad A_s^{12})$, $b_1 = (b^1)$, and $c_1^T = (c^1 \quad c_s^2)$.

Incoming Arcs: Node one has $S+1$ incoming arcs (21) and ($s1$) and their types are $\mathcal{T}_{21}$ =up and $\mathcal{T}_{s1}$ =up for all $s \in \{1,\ldots,S\}$. They all connect the partition graphs p_1 and p_2.

Added Variables: $l_{p_2} \in \Re^{\mathcal{K}_2}$, $\theta_{p_2} \in \Re$, $\psi_{21} \in \Re^{\mathcal{C}_{21}}$, and $\psi_{s1} \in \Re^{\mathcal{C}_{s1}}$. The sources of the incoming arcs are in p_2. Therefore, l and θ are subscripted by p_2.

Added Data: 1 is indexed by (θ_{p_2}, s_1), $\tilde{g}_{p_2}$ is indexed by (θ_{p_2}, l_{p_2}), $\tilde{X}_{12}$ is indexed by (ψ_{21}, l_{p_2}), $\tilde{X}_{1s}$ is indexed by (ψ_{s1}, l_{p_2}), $-I_{12}$ is indexed by (ψ_{21}, x_1), and $-I_{1s}$ is indexed by (ψ_{s1}, x_1) for all $s \in \{1,\ldots,S\}$. Note that $\tilde{g}$ is subscripted by p_2.

Non-negativity: $l_{21} \geq 0$, $l_{s1} \geq 0$ and x_1 is free.

Constraint Types: π_1 is $\geq$, ψ_{21} is $=$, ψ_{s1} is $=$, and θ_{p_2} is $=$.

Formulation ($1 \in \mathcal{N}$): This is a D-W Master problem to the implicit subproblem defined on p_2. It incorporates new columns in a synchronous manner based on a p_2-feasible point for a given value of $(\tilde{\psi}_{21}^T, \tilde{\psi}_{s1}^T)$.

	l_{p_2}	x_1		s_1	
π_1		A_1	$\geq$	b_1	
ψ_{21}	$\tilde{X}_{12}$	$-I_{12}$	$=$	0	
ψ_{s1}	$\tilde{X}_{1s}$	$-I_{1s}$	$=$	0	$\forall s \in \{1,\ldots,S\}$,
θ_{p_2}	$\tilde{g}_{p_2}^T$		$=$	1	
	≥ 0	*free*			
σ_1	0	c_1^T			

Stochastic Subproblem ($2 \in \mathcal{N}$): Node 2 is leftmost and not topmost.

Original Variables: $\pi_2 \in \Re^{\mathcal{R}_2}$ and $x_2 \in \Re^{\mathcal{C}_2}$.

Original Data: $A_2 = (A^{12})$, $b_2 = (b^2)$, and $c_2^T = (0)$.

Incoming Arcs: There are $S+1$ arcs entering node two and their types are $\mathcal{T}_{12} =$ down and $\mathcal{T}_{s2} =$ left. They all lie within p_2.

Added Variables: $w_{12} \in \Re$, $v_{12} \in \Re$, $\lambda_{s2} \in \Re^{K_s}$, $t_{s2} \in \Re$, and $y_{s2} \in \Re^{\mathcal{R}_{s2}}$.

Added Data: $\tilde{\psi}_{21}$ is indexed by (v_{12}, x_2), $-\tilde{\theta}_{21}$ is indexed by (v_{12}, s), $\tilde{\delta}_{21}$ is indexed by (v_{12}, w_{12}), another $\tilde{\delta}_{21}$ is indexed by (σ_2, w_{12}), 1 is indexed by (σ_2, t_{s2}), $\tilde{\gamma}_{2s}$ is indexed by (λ_{s2}, t_{s2}), $\tilde{\Pi}_{2s}$ is indexed by (λ_{s2}, y_{s2}), and $-I_{2s}$ is indexed by (π_2, y_{s2}).

Non-negativity: $x_2 \geq 0$, and y_{s2}, w_{12}, and t_{s2} are free.

Constraint Types: λ_{s2} is $\geq$, v_{12} is $\geq$, and π_2 is $=$.

Formulation ($2 \in \mathcal{N}$): This is a Benders Master program to the subproblems defined over s. Each subproblem adds constraints independently of the others.

	x_2	y_{s2}	w_{12}	t_{s2}		s_2
λ_{s2}		$\tilde{\Pi}_{2s}$		$\tilde{\gamma}_{2s}$	$\geq$	0
v_{12}	$\tilde{\psi}_{21}^T$		$\tilde{\delta}_{21}$		$\geq$	$-\tilde{\theta}_{21}$
π_2	A_2	$-I_{2s}$			$=$	b_2
	≥ 0	*free*	*free*	*free*		
σ_2	0	0	$\tilde{\delta}_{21}$	1		

Stochastic Subproblem ($s \in \mathcal{N}$): Node s is neither topmost nor leftmost.

Original Variables: $\pi_s \in \Re^{\mathcal{R}_s}$ and $x_s \in \Re^{\mathcal{C}_s}$.

Original Data: $A_s = (A^{22})$, $b_s = (0)$, and $c_s^T = (0)$.

Incoming Arcs: Node s has two entering arcs $(1s)$ and $(2s)$ with types $\mathcal{T}_{1s}$ = down, $\mathcal{T}_{2s}$ = right.

Added Variables: $w_{1s} \in \Re$, $v_{1s} \in \Re$, $\omega_{2s} \in \Re$, and $u_{2s} \in \Re$.

Added Data: $\tilde{\psi}_{s1}$ is indexed by (v_{1s}, x_s), $-\tilde{\theta}_{s1}$ is indexed by (v_{1s}, s_s), $\tilde{\delta}_{s1}$ is indexed by (v_{1s}, w_{1s}), $\tilde{\delta}_{s1}$ is indexed by (σ_s, w_{1s}), $\tilde{y}_{s2}$ is indexed by (π_s, u_{2s}), $-\tilde{t}_{s2}$ is indexed by (σ_s, u_{2s}), $\tilde{d}_{s2}$ is indexed by (ω_{1s}, u_{2s}), and another $\tilde{d}_{s2}$ is indexed by (ω_{1s}, s_s).

Non-negativity: $u_{2s} \geq 0$, $x_s \geq 0$, and w_{1s} is free.

Constraint Types: v_{1s} is $\geq$, π_s is $\geq$, and ω_{2s} is $=$.

Formulation ($s \in \mathcal{N}$):

	u_{2s}	x_s	w_{1s}		s_s
v_{1s}		$\tilde{\psi}_{s1}^T$	$\tilde{\delta}_{s1}$	$\geq$	$-\tilde{\theta}_{s1}$
π_s	$\tilde{y}_{s2}$	A_s		$\geq$	0
ω_{1s}	$\tilde{d}_{s2}$			$=$	$\tilde{d}_{s2}$
	≥ 0	≥ 0	*free*		
σ_s	$-\tilde{t}_{s2}$	0	$\tilde{\delta}_{s1}$		

A.4 Dense Example

This final example demonstrates cross splitting on a dense matrix. The communication network has five nodes, one of which has an empty row index set. This is a trick by which we can apply the cross splitting technique to the extent that each subproblem is based on a single coefficient of the constraint matrix. Our starting formulation is

$$\begin{array}{lll} \min\limits_{\substack{x^1\geq 0\\ x^2\geq 0}} & & c^1x^1 + \ c^2x^2 = z \\ \text{s.t.} & \pi^1: & A^{11}x^1 + A^{12}x^2 \geq b^1, \\ & \pi^2: & A^{21}x^1 + A^{22}x^2 \geq b^2. \end{array} \tag{A.4}$$

Dense Problem Description:

$$\mathcal{R} = \pi^1 \cup \pi^2, \quad \mathcal{C} = x^1 \cup x^2,$$

$$A = \begin{pmatrix} A^{11} & A^{12} \\ A^{21} & A^{22} \end{pmatrix} \in \Re^{\mathcal{R}\times\mathcal{C}}, \quad b = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix} \in \Re^{\mathcal{R}}, \quad c^T = \begin{pmatrix} c^1 & c^2 \end{pmatrix} \in \Re^{1\times\mathcal{C}}.$$

Dense Communication Network Description:

$\mathcal{N} = (\{1,2,3,4,5\},$
$\mathcal{A} = \{(12),(21),(13),(31),(14),(41),(15),(51),(23),(32),(45),(54)\})$
$\mathcal{R}_1 = \emptyset, \quad \mathcal{R}_2 = \pi^1, \quad \mathcal{R}_3 = \pi^1, \quad \mathcal{R}_4 = \pi^2, \quad \mathcal{R}_5 = \pi^2,$
$\mathcal{C}_1 = x^1 \cup x^2, \quad \mathcal{C}_2 = x^1, \quad \mathcal{C}_3 = x^2, \quad \mathcal{C}_4 = x^1, \quad \mathcal{C}_5 = x^2,$
$\mathcal{T}_{12} = \mathcal{T}_{13} = \mathcal{T}_{14} = \mathcal{T}_{15} = \text{down}, \quad \mathcal{T}_{21} = \mathcal{T}_{31} = \mathcal{T}_{41} = \mathcal{T}_{51} = \text{up},$
$\mathcal{T}_{23} = \mathcal{T}_{45} = \text{right}, \quad \mathcal{T}_{32} = \mathcal{T}_{54} = \text{left}.$

Dense Incidence Graph Description:

$$h = (\{\sigma, \pi^1, \pi^2, s, x^1, x^2\}, \{(\sigma x^1), (\sigma x^2), (\pi^1 s), (\pi^1 x^1), (\pi^1 x^*), (\pi^2 s), (\pi^2 x^1), (\pi^2 x^2)\}).$$

Dense Arc Index Sets:

$$\mathcal{R}_{23} = \mathcal{R}_{32} = \pi^1, \quad \mathcal{R}_{45} = \mathcal{R}_{54} = \pi^2,$$

$$\mathcal{C}_{12} = \mathcal{C}_{21} = x^1, \quad \mathcal{C}_{13} = \mathcal{C}_{31} = x^2, \quad \mathcal{C}_{14} = \mathcal{C}_{41} = x^1, \quad \mathcal{C}_{15} = \mathcal{C}_{51} = x^2.$$

Dense Partition Graphs: There are three partition graphs p_1, p_2, and p_3 in the communication network of this example. As determined by the ordering of the nodes, p_1 is the parent to both p_2 and p_3:

$$p_1 = (\{1,2,3,4,5\}, \{(12),(21),(13),(31),(14),(41),(15),(51)\}),$$

$$\mathcal{R}_{p_1} = \mathcal{R}, \quad \mathcal{C}_{p_1} = \mathcal{C},$$

$$p_2 = (\{2,3\}, \{(23),(32)\}), \quad \mathcal{R}_{p_2} = \pi^1, \quad \mathcal{C}_{p_2} = \mathcal{C},$$

$$p_3 = (\{4,5\}, \{(45),(54)\}), \quad \mathcal{R}_{p_3} = \pi^2, \quad \mathcal{C}_{p_3} = \mathcal{C}.$$

Therefore, the added variables associated with arcs (21), (31), (41), and (51), which link the child partition nodes $\{2,3,4,5\}$ to the parent partition node $\{1\}$, will be indexed by child partitions, namely p_2 and p_3.

Dense Subproblem ($1 \in \mathcal{N}$): Node one is topmost and leftmost.

Original Variables: $x_1 \in \Re^{\mathcal{C}_1}$ since the row index set for node one is empty.

Original Data: $c_1^T = (c^1 \quad c^2)$.

Incoming Arcs: Node one has four entering arcs with sources in two different child partitions. Arcs (21) and (31) are from p_2 and arcs (41) and (51) are from p_3. Their types are $\mathcal{T}_{12} = \mathcal{T}_{13} = \mathcal{T}_{14} = \mathcal{T}_{15} =$ up.

Added Variables: $\psi_{21} \in \Re^{\mathcal{C}_{21}}$, $\psi_{31} \in \Re^{\mathcal{C}_{31}}$, $\psi_{41} \in \Re^{\mathcal{C}_{41}}$, $\psi_{51} \in \Re^{\mathcal{C}_{51}}$, $\theta_{p_2} \in \Re$, $\theta_{p_3} \in \Re$, $l_{p_2} \in \Re^{K_{p_2}}$, and $l_{p_3} \in \Re^{K_{p_3}}$.

Added Data: 1 is indexed by (θ_{p_2}, s_1), 1 is indexed by (θ_{p_3}, s_1), $\tilde{g}_{p_2}$ is indexed by (θ_{p_2}, l_{p_2}), $\tilde{g}_{p_3}$ is indexed by (θ_{p_3}, l_{p_3}), $\tilde{X}_{12}$ is indexed by (ψ_{21}, l_{p_2}), $\tilde{X}_{13}$ is indexed by (ψ_{31}, l_{p_2}), $\tilde{X}_{14}$ is indexed by (ψ_{41}, l_{p_3}), $\tilde{X}_{15}$ is indexed by (ψ_{51}, l_{p_3}), $-I_{12}$ is indexed by (ψ_{21}, x_1), $-I_{13}$ is indexed by (ψ_{31}, x_1), $-I_{14}$ is indexed by (ψ_{41}, x_1), and $-I_{15}$ is indexed by (ψ_{51}, x_1).

Non-negativity: $l_{p_2} \geq 0$, $l_{p_3} \geq 0$, and x_1 is free.

Constraint Types: ψ_{21}, ψ_{31}, ψ_{41}, ψ_{51}, θ_{p_2}, and θ_{p_3} are all equalities.

Formulation ($1 \in \mathcal{N}$):

	l_{p_2}	l_{p_3}	x_1		s_1
ψ_{21}	$\tilde{X}_{12}$		$-I_{12}$	=	0
ψ_{31}	$\tilde{X}_{13}$		$-I_{13}$	=	0
ψ_{41}		$\tilde{X}_{14}$	$-I_{14}$	=	0
ψ_{51}		$\tilde{X}_{15}$	$-I_{15}$	=	0
θ_{p_2}	$\tilde{g}_{p_2}^T$			=	1
θ_{p_3}		$\tilde{g}_{p_3}^T$		=	1
	≥ 0	≥ 0	*free*		
σ_1	0	0	c_1^T		

Dense Subproblem ($2 \in \mathcal{N}$): Node two is leftmost and not topmost.

Original Variables: $\pi_2 \in \Re^{\mathcal{R}_2}$ and $x_2 \in \Re^{\mathcal{C}_2}$.

Original Data: $A_2 = (A^{11})$, $b_2 = (b^1)$, and $c_2^T = (0)$.

Incoming Arcs: There are two arcs entering this node, (12) and (32), and their types are $\mathcal{T}_{12} =$ down, $\mathcal{T}_{32} =$ left. Arc (12) spans p_1 and p_2 but it is down so no explicit synchronization is necessary.

Added Variables: $\lambda_{32} \in \Re^{K_3}$, $v_{12} \in \Re$, $y_{32} \in \Re^{\mathcal{R}_{32}}$, $w_{12} \in \Re$, and $t_{32} \in \Re$.

Added Data: $\tilde{\psi}_{21}$ is indexed by (v_{12}, x_2), $-\tilde{\theta}_{21}$ is indexed by (v_{12}, s_2), δ_{21} is indexed by (v_{12}, w_{12}), δ_{21} is indexed by (σ_2, w_{12}), another 1 is indexed by (σ_2, t_{32}), $\tilde{\gamma}_{23}$ is indexed by (λ_{32}, t_{32}), $\tilde{\Pi}_{23}$ is indexed by (λ_{32}, y_{32}), and $-I_{23}$ is indexed by (π_2, y_{32}).

Non-negativity: x_2, y_{32}, w_{12}, and t_{32} are all free.

Constraint Types: λ_{32} is $\geq$, v_{12} is $\geq$, and π_2 is $=$.

Formulation ($2 \in \mathcal{N}$):

	x_2	y_{32}	w_{12}	t_{32}		s_2
λ_{32}		$\bar{\Pi}_{23}$		$\bar{\gamma}_{23}$	$\geq$	0
v_{12}	$\bar{\psi}_{21}^T$		δ_{21}		$\geq$	$-\bar{\theta}_{21}$
π_2	A_2	$-I_{23}$			$=$	b_2
	free	*free*	*free*	*free*		
σ_2	0	0	δ_{21}	1		

Dense Subproblem ($3 \in \mathcal{N}$): Node three is neither leftmost nor topmost.

Original Variables: $\pi_3 \in \Re^{\mathcal{R}_3}$ and $x_3 \in \Re^{\mathcal{C}_3}$.

Original Data: $A_3 = (A^{12})$, $b_3 = (0)$, and $c_3^T = (0)$.

Incoming Arcs: There are two incoming arcs to node three, (13) and (23), and their types are $\mathcal{T}_{13} =$ down and $\mathcal{T}_{23} =$ right.

Added Variables: $v_{13} \in \Re$, $\omega_{23} \in \Re$, $u_{23} \in \Re$, and $w_{13} \in \Re$.

Added Data: $\bar{\psi}_{31}$ is indexed by (v_{13}, x_3), $-\bar{\theta}_{31}$ is indexed by (v_{13}, s_3), $\bar{\delta}_{31}$ is indexed by (v_{13}, w_{13}), $\bar{\delta}_{31}$ is indexed by (σ_3, w_{13}), $\bar{y}_{32}$ is indexed by (π_3, u_{23}), $-\bar{t}_{32}$ is indexed by (σ_3, u_{23}), $\bar{d}_{32}$ is indexed by (ω_{23}, u_{23}), and another $\bar{d}_{32}$ is indexed by (ω_{23}, s_3).

Non-negativity: $u_{23} \geq 0$ x_3 is free, and w_{13} is free.

Constraint Types: v_{13} is $\geq$, π_3 is $\geq$, and ω_{23} is $=$.

Formulation ($3 \in \mathcal{N}$):

	u_{23}	x_3	w_{13}		s_3
v_{13}		$\bar{\psi}_{31}^T$	$\bar{\delta}_{31}$	$\geq$	$-\bar{\theta}_{31}$
π_3	$\bar{y}_{32}$	A_3		$\geq$	0
ω_{23}	$\bar{d}_{32}$			$=$	$\bar{d}_{32}$
	≥ 0	*free*	*free*		
σ_3	$-\bar{t}_{32}$	0	$\bar{\delta}_{31}$		

Dense Subproblem ($4 \in \mathcal{N}$): Similar to node two, this node is leftmost and not topmost.

Original Variables: $\pi_4 \in \Re^{\mathcal{R}_4}$, $x_4 \in \Re^{\mathcal{C}_4}$.

Original Data: $A_4 = (A^{21})$, $b_4 = (b^2)$, and $c_4^T = (0)$.

Incoming Arcs: There are two arcs entering node four, and their types are $\mathcal{T}_{14} =$ down and $\mathcal{T}_{54} =$ left.

Added Variables: $\lambda_{54} \in \Re^{\mathcal{K}_5}$, $v_{14} \in \Re$, $y_{54} \in \Re^{\mathcal{R}_{54}}$, $w_{14} \in \Re$, and $t_{54} \in \Re$.

Added Data: $\bar{\psi}_{41}$ is indexed by (v_{14}, x_4), $-\bar{\theta}_{41}$ is indexed by (v_{14}, s_4), $\bar{\delta}_{41}$ is indexed by (v_{14}, w_{14}), $\bar{\delta}_{41}$ is indexed by (σ_4, w_{14}), another 1 is indexed by (σ_4, t_{54}), $\bar{\gamma}_{43}$ is indexed by (λ_{54}, t_{54}), $\bar{\Pi}_{43}$ is indexed by (λ_{54}, y_{54}), and $-I_{43}$ is indexed by (π_4, y_{54}).

Non-negativity: $x_4 \geq 0$, and y_{54}, w_{14}, and t_{54} are free.

Constraint Types: λ_{54} is $\geq$, v_{14} is $\geq$, and π_4 is $=$.

Formulation ($4 \in \mathcal{N}$):

	x_4	y_{54}	w_{14}	t_{54}		s_4
λ_{54}		$\bar{\Pi}_{45}$		$\bar{\gamma}_{45}$	$\geq$	0
v_{14}	$\bar{\psi}_{41}^T$		$\bar{\delta}_{41}$		$\geq$	$-\bar{\theta}_{41}$
π_4	A_4	$-I_{45}$			$=$	b_4
	≥ 0	*free*	*free*	*free*		
σ_4	0	0	$\bar{\delta}_{41}$	1		

Dense Subproblem ($5 \in \mathcal{N}$): Similar to node three, node five is neither topmost nor leftmost.

Original Variables: $\pi_5 \in \Re^{\mathcal{R}_5}$ and $x_5 \in \Re^{\mathcal{C}_5}$.

Original Data: $A_5 = (A^{12})$, $b_5 = (0)$, and $c_5^T = (0)$.

Incoming Arcs: There are two incoming arcs to node three, (15) and (45), and their types are $\mathcal{T}_{15} =$ down and $\mathcal{T}_{45} =$ right.

Added Variables: $v_{15} \in \Re$, $\omega_{45} \in \Re$, $u_{45} \in \Re$, and $w_{15} \in \Re$.

Added Data: $\tilde{\psi}_{51}$ is indexed by (v_{15}, x_5), $-\tilde{\theta}_{51}$ is indexed by (v_{15}, s_5), $\tilde{\delta}_{51}$ is indexed by (v_{15}, w_{15}), $\tilde{\delta}_{51}$ is indexed by (σ_5, w_{15}), $\tilde{y}_{54}$ is indexed by (π_5, u_{45}), $-\tilde{t}_{54}$ is indexed by (σ_5, u_{45}), $\tilde{d}_{54}$ is indexed by (ω_{45}, u_{45}), and another $\tilde{d}_{54}$ is indexed by (ω_{45}, s_5).

Non-negativity: $u_{45} \geq 0$ $x_5 \geq 0$, and w_{15} is free.

Constraint Types: v_{15} is $\geq$, π_5 is $\geq$, and ω_{45} is $=$.

Formulation ($5 \in \mathcal{N}$):

	u_{45}	x_5	w_{15}		s_5
v_{15}		$\bar{\psi}_{51}^T$	$\bar{\delta}_{51}$	$\geq$	$-\bar{\theta}_{51}$
π_5	$\bar{y}_{54}$	A_5		$\geq$	η
ω_{45}	$\bar{d}_{54}$			$=$	$\bar{d}_{54}$
	≥ 0	≥ 0	*free*		
σ_5	$-\bar{t}_{32}$	0	$\bar{\delta}_{51}$		

Appendix B

The Test Problems

For each problem from the test suite, we have produced a bitmap pattern of the nonzeroes in the constraint matrix. The application called SparseDisplay was used with the consent of its creator Irv Lustig.

The three DIET problems were created by the author from an example in Chvátal [Chv83]. They are used primarily for test purposes and are quite small and dense.

The next group of problems are from the standard netlib set.

GROW7, GROW15, and **GROW22** are of unknown nature and origin.

STAIR is also known as DINAMCO, and is an economic model of Mexico due to Alan Manne [Man??].

PILOT4 is an early version of a U.S. energy economic model by George Dantzig and Wesley Winkler.

Finally, the next last group of test problems was first documented in [HL81a], and their descriptions are paraphrased here. Further references are available in the cited publication.

SC205 is an dynamic multisector development planning model.

SCAGR7 and **SCAGR25** are an two versions (respectively 7-period and 25-period) of a large dairy farm expansion planning model.

SCRS8 is a technological assessment model for the transition from fossil to renewable energy resources in the U.S.

SCORPION is a dynamic energy flow model developed for the oil sector of France.

SCSD1, SCSD6, and **SCSD8** are sample problems in the minimal weight design of multistage trusses under a single loading condition.

SCFXM1, SCFXM2, and **SCFXM3** are a production scheduling model (origin unknown).

SCTAP1, SCTAP2, and **SCTAP3** are problems in the optimization of dynamic traffic flow where congestion is modelled explicitly in the flow equations.

Figure B.1: Bitmap of DIET2 (magnification = $\frac{2000}{1000}$).

Figure B.2: Bitmap of DIET3 (magnification = $\frac{2000}{1000}$).

Figure B.3: Bitmap of DIET7 (magnification = $\frac{2000}{1000}$).

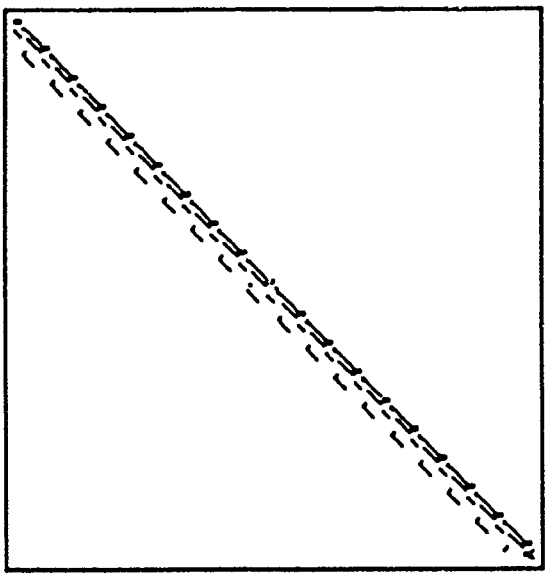

Figure B.4: Bitmap of SC205 (magnification = $\frac{1000}{1000}$).

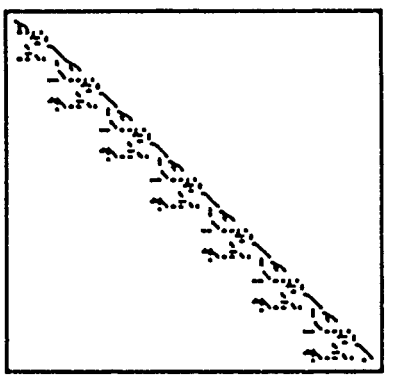

Figure B.5: Bitmap of SCAGR7 (magnification = $\frac{1000}{1000}$).

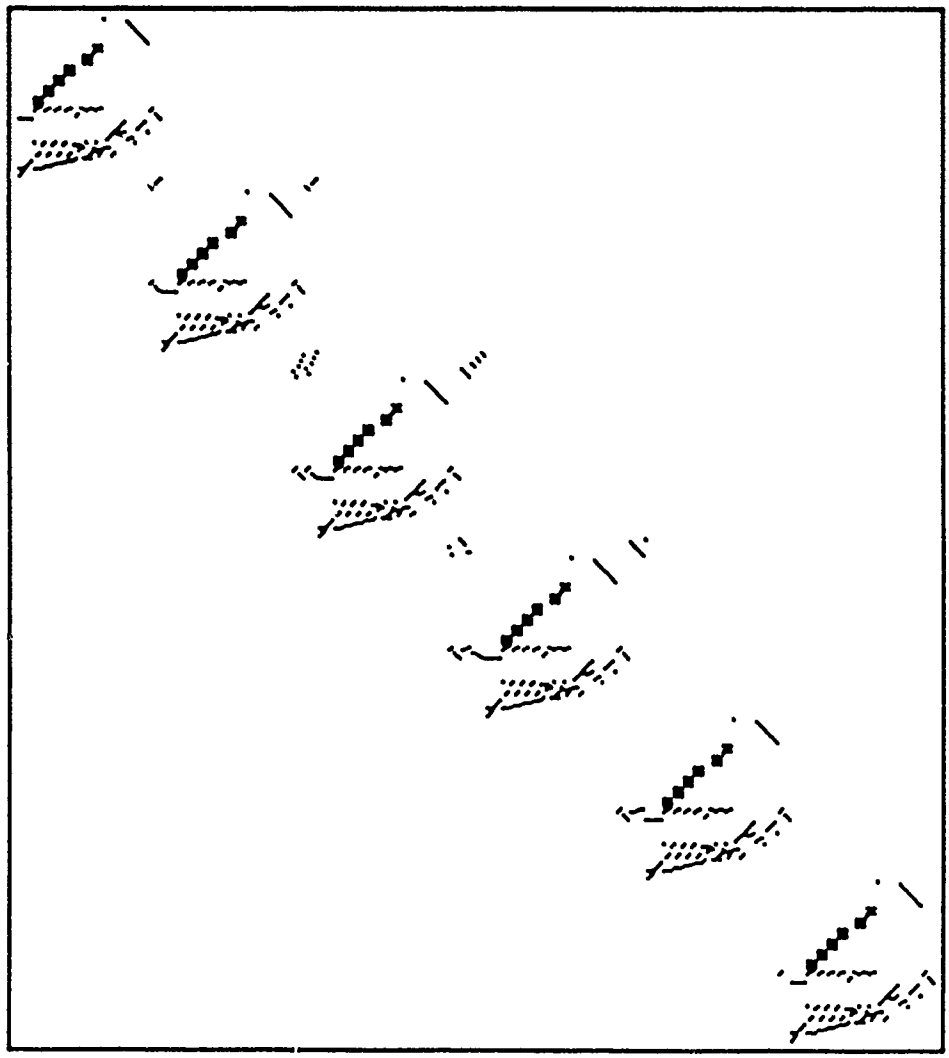

Figure B.6: Bitmap of SCORPION (magnification = $\frac{1000}{1000}$).

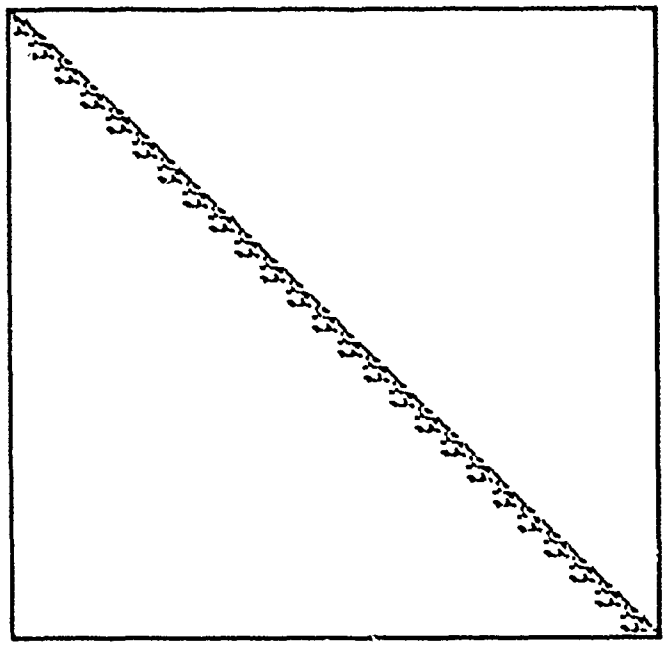

Figure B.7: Bitmap of SCAGR25 (magnification = $\frac{500}{1000}$).

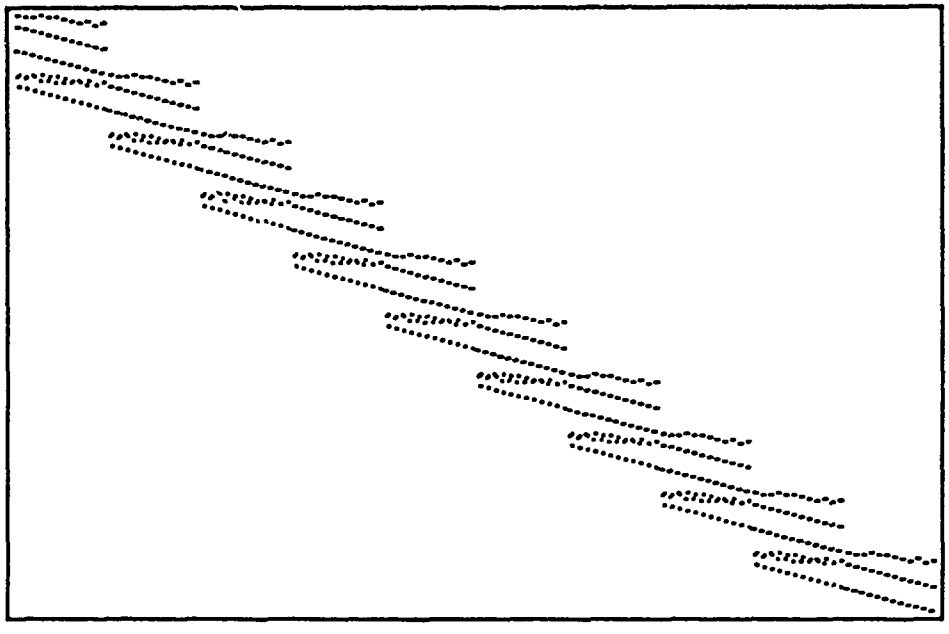

Figure B.8: Bitmap of SCTAP1 (magnification = $\frac{750}{1000}$).

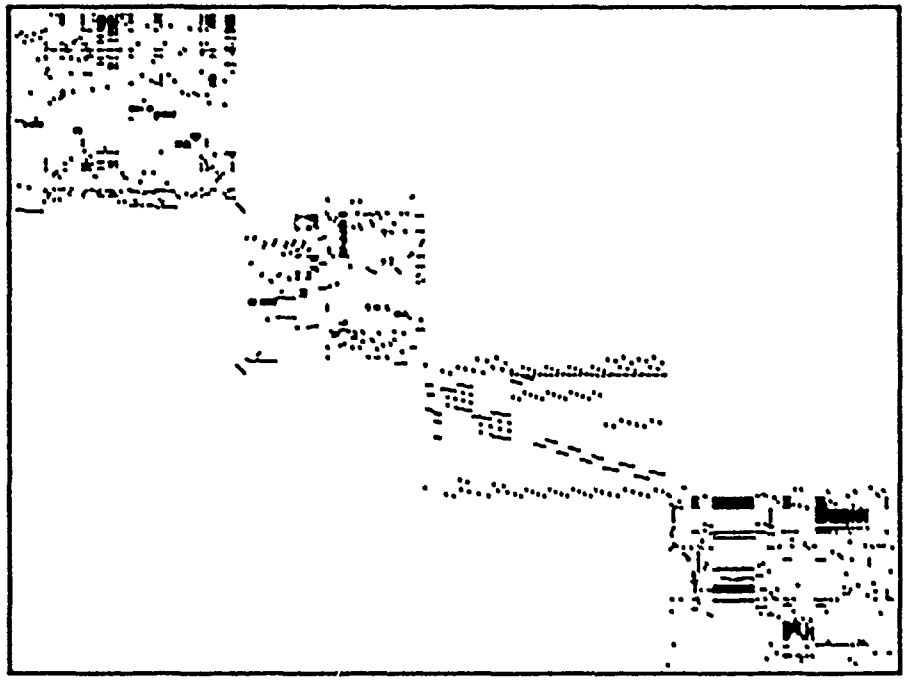

Figure B.9: Bitmap of SCFXM1 (magnification = $\frac{750}{1000}$).

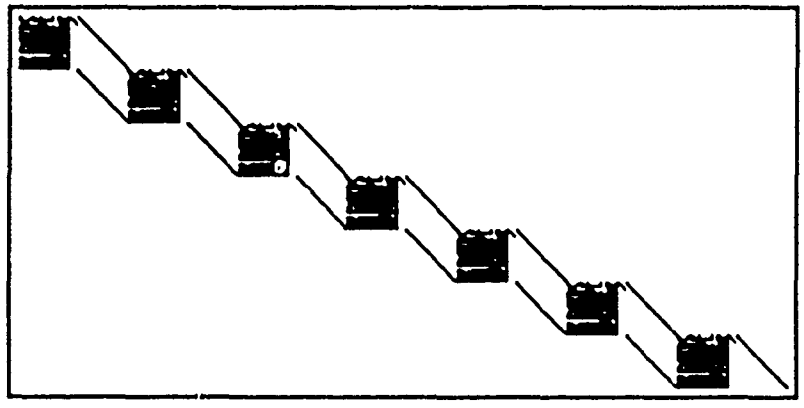

Figure B.10: Bitmap of GROW7 (magnification = $\frac{1000}{1000}$).

Figure B.11: Bitmap of SCSD1 (magnification = $\frac{500}{1000}$).

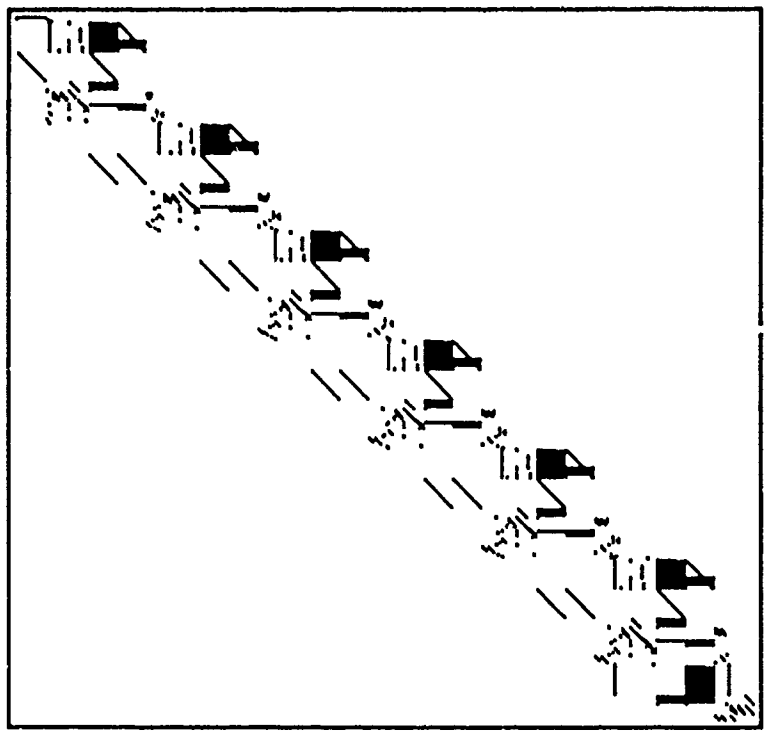

Figure B.12: Bitmap of STAIR (magnification = $\frac{750}{1000}$).

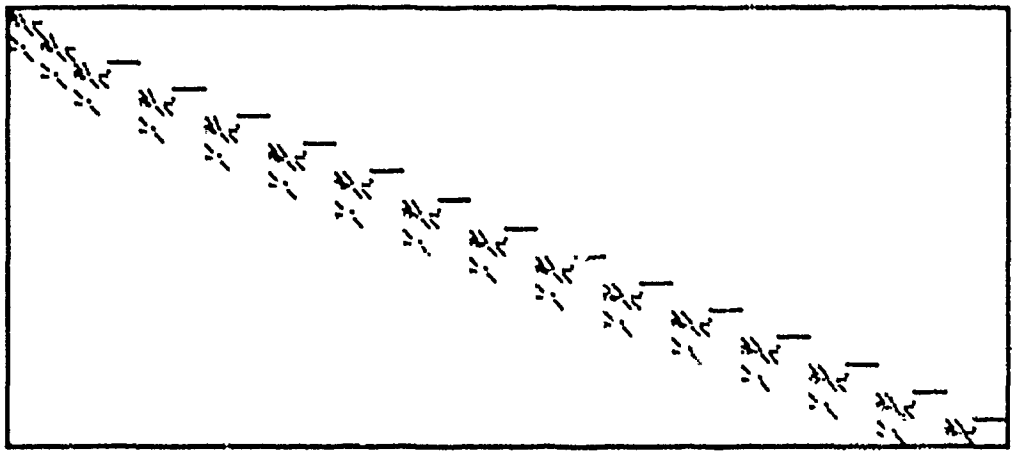

Figure B.13: Bitmap of SCRS8 (magnification = $\frac{333}{1000}$).

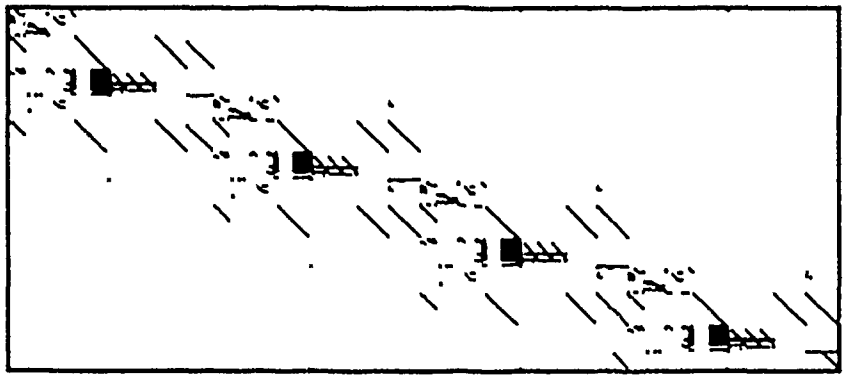

Figure B.14: Bitmap of PILOT4 (magnification = $\frac{333}{1000}$).

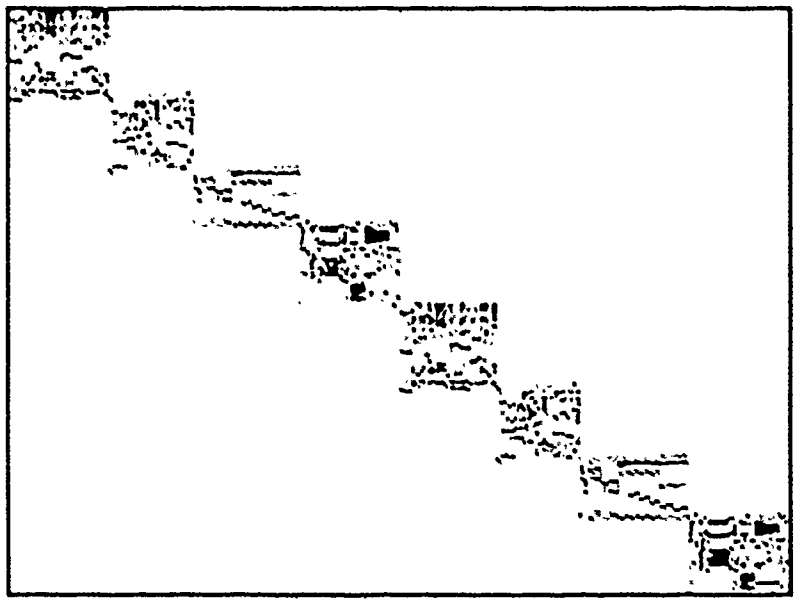

Figure B.15: Bitmap of SCFXM2 (magnification = $\frac{333}{1000}$).

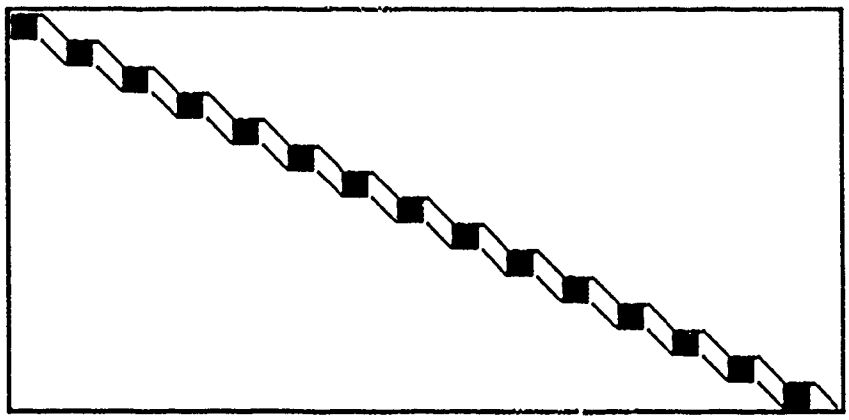

Figure B.16: Bitmap of GROW15 (magnification = $\frac{500}{1000}$).

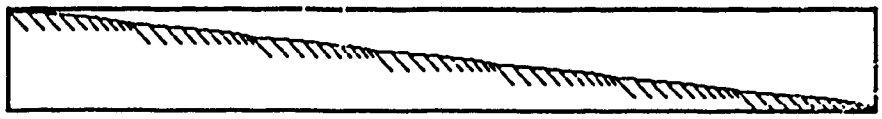

Figure B.17: Bitmap of SCSD6 (magnification = $\frac{250}{1000}$).

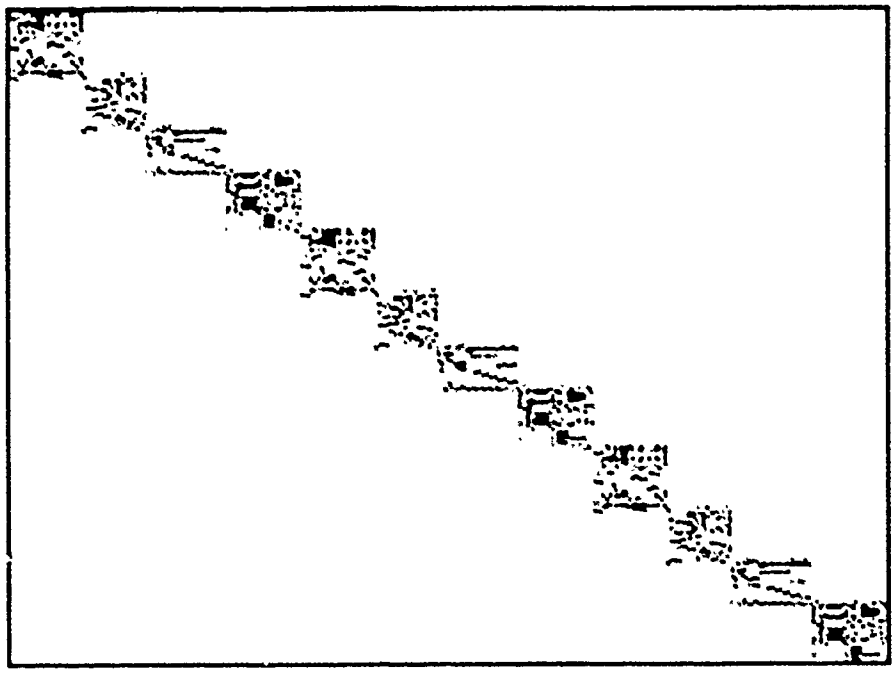

Figure B.18: Bitmap of SCFXM3 (magnification = $\frac{250}{1000}$).

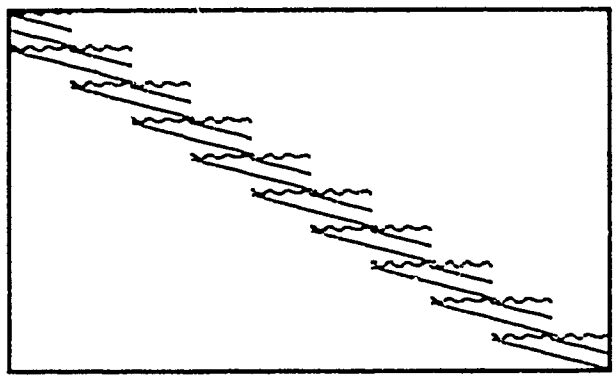

Figure B.19: Bitmap of SCTAP2 (magnification = $\frac{125}{1000}$).

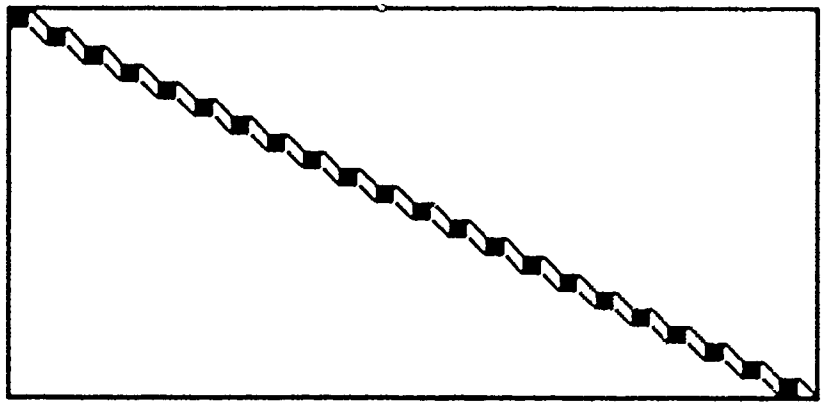

Figure B.20: Bitmap of GROW22 (magnification = $\frac{333}{1000}$).

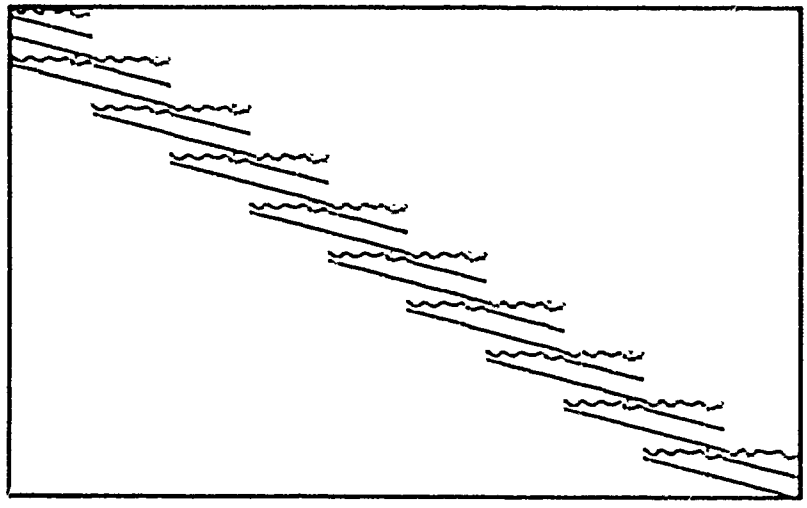

Figure B.21: Bitmap of SCTAP3 (magnification = $\frac{125}{1000}$).

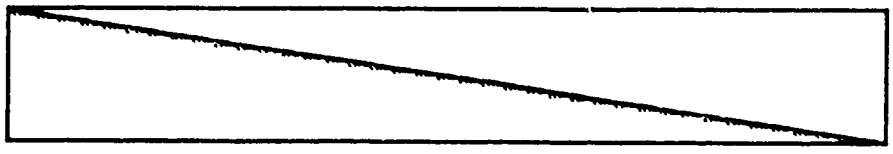

Figure B.22: Bitmap of SCSD8 (magnification = $\frac{125}{1000}$).

Appendix C

Tables

C.1 Constant Number of Subproblems

The first five tables are the supporting data for Figures 4.3, and 4.6–4.9, and the follow are descriptions of thecolumn headings.

NAME The name of a problem from our test suite.

n The number of nodes in the communication network.

p The number of IBM 3090/600E virtual processors.

ITN The total number of simplex method iterations executed on all subproblems.

SLV The total number of solves for all subproblems.

DCPU The cpu time spent for input, solution, and output (micro-seconds).

SCPU The cpu time spent for solution (micro-seconds).

Work The cpu time spent forming and solving subproblems (micro-seconds).

SELP the solution elapsed time (micro-seconds).

OBJTRU the optimal objective value.

Rat the ratio of SCPU/SELP.

Eff the ratio of Rat/p.

Spd the speedup measured as the ratio of the smallest serial time using either MINOS or DECOMP ($p = 1$).

Spin the percentage of solution time not spent forming and solving subproblems; (SCPU-Work)/SCPU.

Name	n	p	Itn	DIv	Slv	Tcpu	Wrk	Cmch	Time	Objective	Pwr	Eff	Spd	Spin
SCAGR7	0	1	93	10	1	402	228	228	230	-0.2331389824331D+07	0.99	99%	1.0	0%
SCAGR7	0	1	93	10	1	407	232	232	235	-0.2331389824331D+07	0.99	99%	1.0	0%
SCAGR7	7	1	452	43	1388	8061	7869	6940	8125	-0.2331085468991D+07	0.97	97%	0.0	12%
SCAGR7	7	1	452	43	1888	8127	7937	6996	8201	-0.2331085468991D+07	0.97	97%	0.0	12%
SCAGR7	7	2	482	47	233	2058	1800	1410	1170	-0.2331341141112D+07	1.54	77%	0.2	22%
SCAGR7	7	2	625	50	341	2646	2386	1956	1484	-0.2331306455361D+07	1.61	80%	0.2	18%
SCAGR7	7	2	442	39	3101	13103	12863	10868	7996	-0.2331006478981D+07	1.61	80%	0.0	16%
SCAGR7	7	3	562	41	306	2620	2316	1772	1278	-0.2330475061066D+07	1.81	60%	0.2	23%
SCAGR7	7	3	467	45	262	2395	2092	1536	1097	-0.2331057799207D+07	1.91	64%	0.2	27%
SCAGR7	7	3	414	41	1750	9703	9403	6992	5791	-0.2331084668216D+07	1.62	54%	0.0	26%
SCAGR7	7	4	331	47	211	2241	1887	1222	1067	-0.2331282252049D+07	1.77	44%	0.2	35%
SCAGR7	7	4	413	42	1717	12782	12437	6840	8930	-0.2322334978334D+07	1.39	35%	0.0	45%
SCAGR7	7	4	394	48	266	2723	2375	1492	1367	-0.2326167666179D+07	1.74	43%	0.2	37%
SCORPION	0	1	139	61	1	1150	785	785	797	0.1878124822738D+04	0.98	98%	1.0	0%
SCORPION	0	1	139	61	1	1150	785	785	793	0.1878124822738D+04	0.99	99%	1.0	0%
SCORPION	6	1	280	126	114	2438	2027	1975	2072	0.1878124822738D+04	0.98	98%	0.4	3%
SCORPION	6	1	280	126	114	2437	2026	1974	2060	0.1878124822738D+04	0.98	98%	0.4	3%
SCORPION	6	2	279	125	130	4233	3774	2116	2337	0.1878124822738D+04	1.61	81%	0.3	44%
SCORPION	6	2	280	126	114	4141	3680	2034	2269	0.1878124822738D+04	1.62	81%	0.3	45%
SCORPION	6	2	280	126	114	4179	3718	2058	2139	0.1878124822738D+04	1.74	87%	0.4	45%
SCORPION	6	3	278	124	132	4467	3959	2149	2334	0.1878124822738D+04	1.70	57%	0.3	46%
SCORPION	6	3	280	126	130	4496	3990	2143	2391	0.1878124822738D+04	1.67	56%	0.3	46%
SCORPION	6	3	280	126	116	4331	3824	2051	2216	0.1878124822738D+04	1.73	58%	0.4	46%
SCORPION	6	4	280	126	148	5297	4751	2221	3113	0.1878124822738D+04	1.53	38%	0.3	53%
SCORPION	6	4	283	129	148	5019	4469	2234	2652	0.1878124822738D+04	1.69	42%	0.3	50%
SCORPION	6	4	276	125	192	5391	4839	2464	3015	0.1878124822738D+04	1.60	40%	0.3	49%
SCAGR25	0	1	475	116	1	4783	4343	4343	4382	-0.1475343306077D+08	0.99	99%	1.0	0%
SCAGR25	0	1	475	116	1	4768	4332	4332	4378	-0.1475343306077D+08	0.99	99%	1.0	0%
SCAGR25	3	1	1317	71	1002	10190	9733	9343	9888	-0.7034688222719D+07	0.98	98%	0.4	4%
SCAGR25	3	1	1317	71	1002	10247	9791	9396	9947	-0.7034688222719D+07	0.98	98%	0.4	4%
SCAGR25	3	2	1113	72	1002	17059	16554	9711	9909	-0.7034689473487D+07	1.67	84%	0.4	41%
SCAGR25	3	2	1120	72	1001	16475	15969	9619	9951	-0.7034689472408D+07	1.60	80%	0.4	40%
SCAGR25	3	2	1128	71	1001	16486	15983	9671	10111	-0.7034651433827D+07	1.58	79%	0.4	39%
SCAGR25	3	3	1009	76	1002	18167	17616	9699	10172	-0.7034651480718D+07	1.73	58%	0.4	45%
SCAGR25	3	3	1023	76	1001	18344	17762	9761	10447	-0.7034688227004D+07	1.70	57%	0.4	45%
SCAGR25	3	3	1012	71	1001	18457	17874	9776	10180	-0.7034651444565D+07	1.76	59%	0.4	45%
SCAGR25	3	4	1024	76	1001	20497	19865	9819	11813	-0.7034651435092D+07	1.68	42%	0.4	51%
SCAGR25	3	4	1058	71	1003	21407	20770	9979	12934	-0.7034651432544D+07	1.61	40%	0.3	52%
SCAGR25	3	4	1034	69	1001	21037	20410	9836	12466	-0.7034689474197D+07	1.64	41%	0.4	52%
SCTAP1	0	1	354	138	1	2292	1888	1888	1926	0.1412250000000D+04	0.98	98%	0.8	0%
SCTAP1	0	1	354	138	1	2301	1897	1897	1914	0.1412250000000D+04	0.99	99%	0.8	0%
SCTAP1	10	1	513	30	163	1987	1531	1448	1576	0.1412250000000D+04	0.97	97%	1.0	5%
SCTAP1	10	1	513	30	163	1995	1539	1454	1577	0.1412250000000D+04	0.98	98%	1.0	6%
SCTAP1	10	2	749	58	255	3009	2504	2197	1396	0.1412250000000D+04	1.79	90%	1.1	12%
SCTAP1	10	2	627	29	204	2640	2130	1830	1228	0.1412250000000D+04	1.73	87%	1.3	14%
SCTAP1	10	2	589	31	232	2690	2184	1892	1219	0.1412250000000D+04	1.79	90%	1.3	13%
SCTAP1	10	3	640	38	212	2839	2291	1887	1086	0.1412250000000D+04	2.11	70%	1.5	18%
SCTAP1	10	3	645	33	231	2969	2417	2003	1078	0.1412250000000D+04	2.24	75%	1.5	17%
SCTAP1	10	3	751	43	378	3824	3274	2823	1490	0.1412250000000D+04	2.20	73%	1.1	14%
SCTAP1	10	4	558	30	237	2965	2362	1915	1029	0.1412250000000D+04	2.30	57%	1.5	19%
SCTAP1	10	4	719	34	331	3806	3207	2616	1433	0.1412250000000D+04	2.24	56%	1.1	18%
SCTAP1	10	4	665	28	253	3653	3057	2136	1506	0.1412250000000D+04	1.90	48%	1.0	30%

Table C.2: Constant number of subproblems.

Name	n	p	Itn	DPiv	Slv	Tcpu	Wrk	Cmch	Time	Objective	Pwr	Eff	Spd	Spin
SCFXM1	0	1	416	154	1	2901	2468	2468	2482	0.1841675902835D+05	0.99	99%	1.0	0%
SCFXM1	0	1	416	154	1	2896	2466	2466	2483	0.1841675902835D+05	0.99	99%	1.0	0%
SCFXM1	4	1	2004	518	313	6587	6107	5981	6254	0.1841675902835D+05	0.98	98%	0.4	2%
SCFXM1	4	1	2004	518	313	6575	6095	5967	6175	0.1841675902835D+05	0.99	99%	0.4	2%
SCFXM1	4	2	2023	505	332	11005	10475	6246	6191	0.1841675902835D+05	1.69	85%	0.4	40%
SCFXM1	4	2	2016	510	324	10947	10415	6189	6193	0.1841675902835D+05	1.68	84%	0.4	41%
SCFXM1	4	2	2031	513	356	11299	10765	6410	6223	0.1841675902835D+05	1.73	86%	0.4	40%
SCFXM1	4	3	2374	571	422	13956	13379	7585	7496	0.1841675902835D+05	1.78	59%	0.3	43%
SCFXM1	4	3	2011	508	362	11872	11299	6409	6379	0.1841675902835D+05	1.77	59%	0.4	43%
SCFXM1	4	3	1972	500	340	11636	11062	6163	6339	0.1841675902835D+05	1.75	58%	0.4	44%
SCFXM1	4	4	2265	527	413	14509	13890	7254	8093	0.1841675902835D+05	1.72	43%	0.3	48%
SCFXM1	4	4	1975	491	354	12547	11922	6308	7089	0.1841675902835D+05	1.68	42%	0.4	47%
SCFXM1	4	4	1486	403	282	9702	9079	4774	5180	0.1845102437697D+05	1.75	44%	0.5	47%
GROW7	0	1	190	18	1	1477	1099	1099	1115	-0.4778781181471D+08	0.99	99%	0.8	0%
GROW7	0	1	190	18	1	1488	1101	1101	1111	-0.4778781181471D+08	0.99	99%	0.8	0%
GROW7	7	1	185	2	97	1289	867	814	893	-0.4778781181471D+08	0.97	97%	1.0	6%
GROW7	7	1	185	2	97	1300	877	821	901	-0.4778781181471D+08	0.97	97%	1.0	6%
GROW7	7	2	208	2	144	1804	1331	1110	766	-0.4778781181471D+08	1.74	87%	1.2	17%
GROW7	7	2	218	1	115	1688	1212	987	710	-0.4778781181471D+08	1.71	85%	1.3	19%
GROW7	7	2	119	2	74	1222	748	583	458	-0.4778781181471D+08	1.63	82%	1.9	22%
GROW7	7	3	208	2	163	2074	1557	1217	800	-0.4778781181471D+08	1.95	65%	1.1	22%
GROW7	7	3	210	2	167	2170	1652	1264	861	-0.4778781181471D+08	1.92	64%	1.0	23%
GROW7	7	3	214	3	183	2228	1708	1352	890	-0.4778780861421D+08	1.92	64%	1.0	21%
GROW7	7	4	200	2	170	2276	1717	1270	896	-0.4778781181471D+08	1.92	48%	1.0	26%
GROW7	7	4	215	2	170	2203	1640	1293	796	-0.4778781181471D+08	2.06	52%	1.1	21%
GROW7	7	4	195	2	166	2417	1854	1228	1091	-0.4778781181471D+08	1.70	42%	0.8	34%
SCSD1	0	1	206	182	1	1383	903	903	909	0.8666366674333D+01	0.99	99%	1.0	0%
SCSD1	0	1	206	182	1	1381	899	899	908	0.8666666674333D+01	0.99	99%	1.0	0%
SCSD1	3	1	857	653	31	2430	1901	1882	1922	0.8666666674650D+01	0.99	99%	0.5	1%
SCSD1	3	1	857	653	31	2430	1897	1878	1920	0.8666666674650D+01	0.99	99%	0.5	1%
SCSD1	3	2	935	657	49	3272	2689	2101	1438	0.8666666674333D+01	1.87	93%	0.6	22%
SCSD1	3	2	961	679	51	3329	2743	2164	1458	0.8666666674333D+01	1.88	94%	0.6	21%
SCSD1	3	2	948	671	53	3330	2747	2137	1462	0.8666666674334D+01	1.88	94%	0.6	22%
SCSD1	3	3	677	517	25	3001	2374	1476	1127	0.8666666674333D+01	2.11	70%	0.8	38%
SCSD1	3	3	819	628	60	3803	3178	1973	1496	0.8666666674333D+01	2.12	71%	0.6	38%
SCSD1	3	3	1060	771	62	4383	3757	2442	1723	0.8666666674334D+01	2.18	73%	0.5	35%
SCSD1	3	4	1032	744	73	4382	3715	2419	1561	0.8666666674333D+01	2.38	59%	0.6	35%
SCSD1	3	4	963	713	56	4352	3682	2230	1741	0.8666666674333D+01	2.11	53%	0.5	39%
SCSD1	3	4	589	438	42	2914	2241	1390	1045	0.8666666674333D+01	2.14	54%	0.9	38%
STAIR	0	1	473	36	1	6728	6196	6196	6271	-0.2512669511930D+03	0.99	99%	0.5	0%
STAIR	0	1	473	36	1	6693	6166	6166	6224	-0.2512669511930D+03	0.99	99%	0.5	0%
STAIR	6	1	240	12	286	3870	3292	3175	3360	-0.2087999900000D+03	0.98	98%	1.0	4%
STAIR	6	1	240	12	286	3914	3333	3215	3403	-0.2087999900000D+03	0.98	98%	1.0	4%
STAIR	6	2	228	12	301	6542	5909	3424	3453	-0.2087999900000D+03	1.71	86%	1.0	42%
STAIR	6	2	229	12	292	6525	5896	3370	3508	-0.2087999900000D+03	1.68	84%	1.0	43%
STAIR	6	2	232	12	291	6484	5851	3364	3511	-0.2087999900000D+03	1.67	83%	1.0	43%
STAIR	6	3	216	12	294	6983	6310	3386	3710	-0.2087999900000D+03	1.70	57%	0.9	46%
STAIR	6	3	228	12	288	7066	6393	3310	3723	-0.2087999900000D+03	1.72	57%	0.9	48%
STAIR	6	3	214	12	294	7204	6531	3385	3730	-0.2087999900000D+03	1.75	58%	0.9	48%
STAIR	6	4	214	12	290	7752	7023	3376	4035	-0.2087999900000D+03	1.74	44%	0.8	52%
STAIR	6	4	214	12	289	7484	6761	3362	3808	-0.2087999900000D+03	1.78	44%	0.9	50%
STAIR	6	4	214	12	287	7701	6977	3329	4083	-0.2087999900000D+03	1.71	43%	0.8	52%

Table C.3: Constant number of subproblems.

Name	n	p	Itn	DPiv	Slv	Tcpu	Wrk	Cmch	Time	Objective	Pwr	Eff	Spd	Spin
SCRS8	0	1	861	319	1	9207	8507	8507	8656	0.9042969538008D+03	0.98	98%	0.3	0%
SCRS8	0	1	861	319	1	9211	8511	8511	8585	0.9042969538008D+03	0.99	99%	0.3	0%
SCRS8	7	1	823	235	99	3330	2555	2499	2595	0.9042969538008D+03	0.98	98%	1.0	2%
SCRS8	7	1	823	235	99	3316	2544	2490	2583	0.9042969538008D+03	0.98	98%	1.0	2%
SCRS8	7	2	788	232	131	4766	3934	2644	2182	0.9042969538008D+03	1.80	90%	1.2	33%
SCRS8	7	2	787	231	130	4720	3893	2615	2116	0.9042969538008D+03	1.84	92%	1.2	33%
SCRS8	7	2	801	233	118	4680	3857	2568	2122	0.9042969538008D+03	1.82	91%	1.2	33%
SCRS8	7	3	805	234	145	5047	4180	2759	2123	0.9042969538008D+03	1.97	66%	1.2	34%
SCRS8	7	3	790	233	144	5103	4225	2751	2103	0.9042969538008D+03	2.01	67%	1.2	35%
SCRS8	7	3	799	234	126	4911	4042	2615	2078	0.9042969538008D+03	1.95	65%	1.2	35%
SCRS8	7	4	806	235	155	5512	4590	2889	2156	0.9042969538008D+03	2.13	53%	1.2	37%
SCRS8	7	4	806	234	157	5379	4465	2856	2138	0.9042969538008D+03	2.09	52%	1.2	36%
SCRS8	7	4	807	235	148	5510	4594	2850	2244	0.9042969538008D+03	2.05	51%	1.2	38%
PILOT4	0	1	3730	1111	1	51213	50417	50417	51028	-0.2581016628137D+04	0.99	99%	0.3	0%
PILOT4	0	1	3730	1111	1	51321	50535	50535	51127	-0.2581016628137D+04	0.99	99%	0.3	0%
PILOT4	4	1	2542	178	420	16777	15951	15636	16277	-0.7319905016480D+12	0.98	98%	1.0	2%
PILOT4	4	1	2542	178	420	16807	15980	15663	16596	-0.7319905016480D+12	0.96	96%	1.0	2%
PILOT4	4	2	3084	1168	407	######	######	88612	90094	-0.4643218961124D+15	1.86	93%	0.2	47%
PILOT4	4	2	2345	179	423	29526	28616	15825	16272	-0.7319896083688D+12	1.76	88%	1.0	45%
PILOT4	4	2	467	161	58	5701	4788	3268	2659	-0.3550972527810D+12	1.80	90%	6.1	32%
PILOT4	4	3	829	177	154	12601	11676	6378	6009	-0.1349535969926D+15	1.94	65%	2.7	45%
PILOT4	4	3	2334	177	422	32360	31379	16217	17092	-0.7319897177463D+12	1.84	61%	1.0	48%
PILOT4	4	3	2334	177	422	31835	30884	16064	17392	-0.7319897177463D+12	1.78	59%	0.9	48%
PILOT4	4	4	455	161	61	6599	5583	3321	2564	-0.1704946512274D+13	2.18	54%	6.3	41%
PILOT4	4	4	2329	175	422	33517	32501	16089	18072	-0.7168499466778D+12	1.80	45%	0.9	50%
PILOT4	4	4	455	161	58	6778	5801	3323	2979	-0.3026879763031D+13	1.95	49%	5.5	43%
SCFXM2	0	1	833	292	1	10257	9439	9439	11558	0.3666026156500D+05	0.82	82%	0.8	0%
SCFXM2	0	1	833	292	1	10214	9407	9407	9511	0.3666026156500D+05	0.99	99%	1.0	0%
SCFXM2	8	1	5789	991	678	16627	15752	15423	16273	0.3666029249815D+05	0.97	97%	0.6	2%
SCFXM2	8	1	5789	991	678	16574	15705	15376	15951	0.3666029249815D+05	0.98	98%	0.6	2%
SCFXM2	8	2	6220	1062	856	19118	18191	17652	9802	0.3666026466566D+05	1.86	93%	1.0	3%
SCFXM2	8	2	6054	1071	814	18534	17603	17083	9117	0.3666026790764D+05	1.93	97%	1.0	3%
SCFXM2	8	2	6460	1157	930	20319	19397	18459	10017	0.3666026156500D+05	1.94	97%	0.9	5%
SCFXM2	8	3	5976	1043	927	20953	19983	17757	7807	0.3666026378162D+05	2.56	85%	1.2	11%
SCFXM2	8	3	6024	1015	871	20797	19819	17550	7396	0.3666028795628D+05	2.68	89%	1.3	11%
SCFXM2	8	3	6546	1217	1234	25213	24241	20509	9325	0.3666026156500D+05	2.60	87%	1.0	15%
SCFXM2	8	4	5514	1053	1029	22702	21678	17861	8211	0.3666026167255D+05	2.64	66%	1.2	18%
SCFXM2	8	4	5443	1052	1000	22950	21928	17641	7702	0.3666027096909D+05	2.85	71%	1.2	20%
SCFXM2	8	4	4862	945	858	20527	19471	15608	7469	0.3690368120357D+05	2.61	65%	1.3	20%
GROW15	0	1	539	40	1	6597	5869	5869	5925	-0.1068709412936D+09	0.99	99%	0.5	0%
GROW15	0	1	539	40	1	6709	5979	5979	6060	-0.1068709412936D+09	0.99	99%	0.5	0%
GROW15	15	1	589	1	307	3610	2810	2653	2890	-0.1068709412936D+09	0.97	97%	1.0	6%
GROW15	15	1	589	1	307	3591	2796	2639	2879	-0.1068709412936D+09	0.97	97%	1.0	6%
GROW15	15	2	899	8	644	6600	5754	5104	3217	-0.1068709412936D+09	1.79	89%	0.9	11%
GROW15	15	2	1169	7	328	5232	4385	3904	2414	-0.1068709412936D+09	1.82	91%	1.2	11%
GROW15	15	2	959	6	411	5212	4363	3847	2419	-0.1068709412936D+09	1.80	90%	1.2	12%
GROW15	15	3	720	2	363	4674	3783	3234	1711	-0.1068709412936D+09	2.21	74%	1.7	15%
GROW15	15	3	1073	1	4790	31369	30477	26108	14349	-0.1068709412936D+09	2.12	71%	0.2	14%
GROW15	15	3	1263	6	716	8018	7127	6244	2998	-0.1068709412936D+09	2.38	79%	1.0	12%
GROW15	15	4	944	4	680	7661	6723	5618	2629	-0.1068709412936D+09	2.56	64%	1.1	16%
GROW15	15	4	1100	4	498	6319	5385	4707	2165	-0.1068709412936D+09	2.49	62%	1.3	13%
GROW15	15	4	1066	3	499	6623	5689	4801	2280	-0.1068709412936D+09	2.50	62%	1.3	16%

Table C.4: Constant number of subproblems.

Name	n	p	Itn	DPiv	Slv	Tcpu	Wrk	Cmch	Time	Objective	Pwr	Eff	Spd	Spin
SCSD6	0	1	520	356	1	3214	2389	2389	2409	0.5050000007714D+02	0.99	99%	1.0	0%
SCSD6	0	1	520	356	1	3209	2386	2386	2435	0.5050000007714D+02	0.98	98%	1.0	0%
SCSD6	7	1	2050	1273	107	3806	2913	2858	2960	0.5050000007576D+02	0.98	98%	0.8	2%
SCSD6	7	1	2050	1273	107	3831	2933	2877	2983	0.5050000007576D+02	0.98	98%	0.8	2%
SCSD6	7	2	2379	1480	137	4918	3970	3494	2101	0.5050000008356D+02	1.89	94%	1.1	12%
SCSD6	7	2	2051	1265	111	4354	3407	2944	1813	0.5050000007375D+02	1.88	94%	1.3	14%
SCSD6	7	2	1933	1226	89	4086	3139	2676	1659	0.5050000008402D+02	1.89	95%	1.5	15%
SCSD6	7	3	2329	1450	167	5196	4206	3553	1718	0.5050000008919D+02	2.45	82%	1.4	16%
SCSD6	7	3	2467	1583	174	5298	4304	3759	1735	0.5050000008056D+02	2.48	83%	1.4	13%
SCSD6	7	3	2666	1625	162	5556	4564	3934	1849	0.5050000006985D+02	2.47	82%	1.3	14%
SCSD6	7	4	2425	1517	166	5848	4815	3697	2012	0.5050000007355D+02	2.39	60%	1.2	23%
SCSD6	7	4	2380	1494	230	5957	4918	4018	1822	0.5050000007633D+02	2.70	67%	1.3	18%
SCSD6	7	4	2336	1480	217	5653	4619	3835	1727	0.5050000007620D+02	2.67	67%	1.4	17%
SCFXM3	0	1	1252	422	1	21815	20526	20626	20891	0.5490125454975D+05	0.99	99%	1.0	0%
SCFXM3	0	1	1252	422	1	21995	20818	20818	23007	0.5490125454975D+05	0.90	90%	0.9	0%
SCFXM3	12	1	11525	1805	1282	34435	33172	32565	33854	0.5490130154424D+05	0.98	98%	0.6	2%
SCFXM3	12	1	11525	1805	1282	34661	33379	32762	35018	0.5490130154424D+05	0.95	95%	0.6	2%
SCFXM3	12	2	10285	1659	1296	32072	30749	30072	15776	0.5490128935557D+05	1.95	97%	1.3	2%
SCFXM3	12	2	13711	2106	1760	43173	41838	40905	22778	0.5490131599008D+05	1.84	92%	0.9	2%
SCFXM3	12	2	10049	1610	1280	31560	30234	29336	17237	0.5490131076105D+05	1.75	88%	1.2	3%
SCFXM3	12	3	10675	1715	1544	35177	33810	32859	11796	0.5490130150784D+05	2.87	96%	1.8	3%
SCFXM3	12	3	9672	1546	1335	31359	30002	29092	10542	0.5490128591737D+05	2.85	95%	2.0	3%
SCFXM3	12	3	9399	1644	1339	30203	28846	28031	10289	0.5490129861552D+05	2.80	93%	2.0	3%
SCFXM3	12	4	10424	1700	1711	37914	36495	33695	10347	0.5490133831142D+05	3.53	88%	2.0	8%
SCFXM3	12	4	10504	1729	1864	38979	37546	34656	11305	0.5490128849170D+05	3.32	83%	1.8	8%
SCFXM3	12	4	9153	1558	1630	33829	32415	30018	9119	0.5490127267199D+05	3.55	89%	2.3	7%
SCTAP2	0	1	1529	811	1	30211	28847	28847	29096	0.1724807142857D+04	0.99	99%	0.1	0%
SCTAP2	0	1	1529	811	1	30170	28802	28802	28994	0.1724807142857D+04	0.99	99%	0.1	0%
SCTAP2	10	1	809	126	154	5252	3778	3695	3844	0.1724807142857D+04	0.98	98%	1.0	2%
SCTAP2	10	1	809	126	154	5239	3766	3687	3823	0.1724807142857D+04	0.99	99%	1.0	2%
SCTAP2	10	2	885	115	217	6747	5220	4558	2760	0.1724807142857D+04	1.89	95%	1.4	13%
SCTAP2	10	2	985	135	274	7541	5996	5329	3191	0.1724807142857D+04	1.88	94%	1.2	11%
SCTAP2	10	2	907	111	193	6588	5058	4411	2715	0.1724807142857D+04	1.86	93%	1.4	13%
SCTAP2	10	3	1053	118	266	7943	6366	5594	2511	0.1724807142857D+04	2.54	85%	1.5	12%
SCTAP2	10	3	974	111	218	7151	5576	4875	2246	0.1724807142857D+04	2.48	83%	1.7	13%
SCTAP2	10	3	917	126	253	7304	5734	5023	2326	0.1724807142857D+04	2.47	82%	1.6	12%
SCTAP2	10	4	972	116	276	7889	6263	5369	2134	0.1724807142857D+04	2.93	73%	1.8	14%
SCTAP2	10	4	982	126	271	8029	6410	5446	2326	0.1724807142857D+04	2.76	69%	1.6	15%
SCTAP2	10	4	1082	142	321	8979	7350	6329	2581	0.1724807142857D+04	2.85	71%	1.5	14%
GROW22	0	1	921	100	1	18647	17611	17611	17772	-0.1608343364826D+09	0.99	99%	0.3	0%
GROW22	0	1	921	100	1	18728	17687	17687	17830	-0.1608343364826D+09	0.99	99%	0.3	0%
GROW22	22	1	825	2	519	5729	4597	4341	4706	-0.1608343364826D+09	0.98	98%	1.0	6%
GROW22	22	1	825	2	519	5738	4611	4350	4718	-0.1608343364826D+09	0.98	98%	1.0	6%
GROW22	22	2	883	0	659	7114	5933	5207	3466	-0.1608343364826D+09	1.71	86%	1.4	12%
GROW22	22	2	929	1	742	7753	6572	5769	3725	-0.1608343364826D+09	1.76	88%	1.3	12%
GROW22	22	2	952	2	743	7961	6779	5957	4008	-0.1608343364825D+09	1.69	85%	1.2	12%
GROW22	22	3	917	3	727	8049	6827	5720	3242	-0.1608343364826D+09	2.11	70%	1.5	16%
GROW22	22	3	1134	1	573	7538	6311	5426	2869	-0.1608343364826D+09	2.20	73%	1.6	14%
GROW22	22	3	772	3	726	7959	6728	5482	3378	-0.1608343364826D+09	1.99	66%	1.4	19%
GROW22	22	4	1104	0	626	7885	6611	5668	2678	-0.1608343364826D+09	2.47	62%	1.8	14%
GROW22	22	4	939	3	592	7482	6207	5187	2508	-0.1608343364826D+09	2.47	62%	1.9	16%
GROW22	22	4	961	0	560	7218	5946	5067	2280	-0.1608343364826D+09	2.61	65%	2.1	15%

Table C.5: Constant number of subproblems.

Name	n	p	Itn	DPiv	Slv	Tcpu	Wrk	Cmch	Time	Objective	Pwr	Eff	Spd	Spin
SCTAP3	0	1	1696	1012	1	43964	42160	42160	42570	0.1424000000000D+04	0.99	99%	0.1	0%
SCTAP3	0	1	1695	1012	1	43929	42119	42119	42423	0.1424000000000D+04	0.99	99%	0.1	0%
SCTAP3	10	1	677	87	134	5958	4008	3937	4065	0.1424000000000D+04	0.99	99%	1.0	2%
SCTAP3	10	1	677	87	134	5939	3997	3928	4054	0.1424000000000D+04	0.99	99%	1.0	2%
SCTAP3	10	2	707	77	144	6892	4907	4131	2601	0.1424000000000D+04	1.89	94%	1.6	16%
SCTAP3	10	2	750	82	143	7075	5084	4299	2724	0.1424000000000D+04	1.87	93%	1.5	15%
SCTAP3	10	2	759	82	140	7065	5073	4302	2667	0.1424000000000D+04	1.90	95%	1.5	15%
SCTAP3	10	3	742	81	223	8129	6093	5108	2535	0.1424000000000D+04	2.40	80%	1.6	16%
SCTAP3	10	3	769	85	221	8264	6224	5275	2527	0.1424000000000D+04	2.46	82%	1.6	15%
SCTAP3	10	3	759	86	239	8357	6315	5354	2574	0.1424000000000D+04	2.45	82%	1.6	15%
SCTAP3	10	4	750	85	259	9322	7241	5556	2960	0.1424000000000D+04	2.45	61%	1.4	23%
SCTAP3	10	4	737	83	201	8315	6228	4948	2376	0.1424000000000D+04	2.62	66%	1.7	21%
SCTAP3	10	4	744	80	239	8570	6487	5356	2291	0.1424000000000D+04	2.83	71%	1.8	17%
SCSD8	0	1	1174	817	1	15722	14109	14109	14225	0.9049999999255D+03	0.99	99%	0.7	0%
SCSD8	0	1	1174	817	1	15699	14075	14075	14179	0.9049999999255D+03	0.99	99%	0.7	0%
SCSD8	7	1	4381	3174	102	11450	9709	9657	9802	0.9049999999255D+03	0.99	99%	1.0	1%
SCSD8	7	1	4381	3174	102	11463	9713	9661	9831	0.9049999999255D+03	0.99	99%	1.0	1%
SCSD8	7	2	4149	3055	150	13104	11303	9322	5870	0.9049999999255D+03	1.93	96%	1.7	18%
SCSD8	7	2	4326	3173	150	13676	11881	9962	6140	0.9049999999255D+03	1.94	97%	1.6	16%
SCSD8	7	2	4210	3099	150	13310	11516	9525	5971	0.9049999999255D+03	1.93	96%	1.6	17%
SCSD8	7	3	5091	3702	157	15742	13902	11655	5524	0.9049999999255D+03	2.52	84%	1.8	16%
SCSD8	7	3	5105	3756	204	16086	14250	11997	5675	0.9049999999254D+03	2.51	84%	1.7	16%
SCSD8	7	3	3840	2816	176	13444	11605	9228	4832	0.9049999999454D+03	2.40	80%	2.0	20%
SCSD8	7	4	4443	3323	183	14744	12865	10388	4584	0.9049999999257D+03	2.81	70%	2.1	19%
SCSD8	7	4	5638	4068	215	17828	15936	13352	5344	0.9049999999255D+03	2.98	75%	1.8	16%
SCSD8	7	4	5022	3707	251	17046	15163	12069	5508	0.9049999999254D+03	2.75	69%	1.8	20%

Table C.6: Constant number of subproblems.

C.2 Varying the Number of Subproblems

The next three tables support Figures 4.10 – 4.12 and following are descriptions of their column headings.

NAME The name of a problem from our test suite.

N The number of subproblems.

DECOMP/1 The amount of work, time, or speedup required to solve SCSD8/*N* with one processor.

DECOMP/2 The amount of work, time, or speedup required to solve SCSD8/*N* with two processors.

DECOMP/3 The amount of work, time, or speedup required to solve SCSD8/*N* with three processors.

DECOMP/4 The amount of work, time, or speedup required to solve SCSD8/*N* with four processors.

Work is in CPU seconds, Time is in seconds, and Speedup is dimensionless.

Name	N	DECOMP/1	DECOMP/2	DECOMP/3	DECOMP/4
SCSD8	2	14.1	20.9	23.7	23.5
	4	11.6	17.1	18.3	17.7
	6	10.9	14.0	16.8	15.9
	8	12.3	14.8	21.7	20.3
	10	9.7	12.2	11.5	13.7
	12	10.1	12.2	11.4	15.4
	14	9.4	13.8	16.5	14.2
	16	14.5	12.8	13.6	14.5
	18	11.5	14.4	13.6	18.2
	20	15.8	16.7	12.8	14.7
	22	14.3	16.4	18.1	16.5
	24	14.7	18.1	15.5	18.4
	26	18.1	22.5	22.4	15.9
	28	21.3	22.0	17.1	20.8
	30	20.3	26.4	20.9	30.7
	32	24.1	21.8	26.7	19.8
	34	22.6	25.6	26.3	23.8
	36	19.9	22.5	28.5	25.2
	38	24.7	30.6	21.0	21.1

Table C.7: Work for a varying number of subproblems.

Name	N	DECOMP/1	DECOMP/2	DECOMP/3	DECOMP/4
SCSD8	2	13.5	11.7	8.5	8.1
	4	10.9	8.3	6.1	5.6
	6	10.3	6.8	5.8	4.6
	8	11.7	7.2	7.1	5.3
	10	9.1	5.8	3.9	3.4
	12	9.4	5.8	3.7	3.8
	14	8.7	6.6	5.4	3.7
	16	13.9	6.1	4.4	3.6
	18	10.9	6.9	4.3	4.6
	20	15.2	8.0	5.8	3.6
	22	13.7	7.9	5.9	4.0
	24	14.0	8.8	5.0	4.5
	26	17.4	11.0	7.3	3.9
	28	20.7	10.7	5.6	5.1
	30	19.6	12.9	6.8	7.7
	32	23.4	10.7	8.8	5.0
	34	22.0	12.5	8.6	5.9
	36	19.3	11.0	9.3	6.2
	38	24.0	15.1	6.8	5.2

Table C.8: Time for a varying number of subproblems.

Name	N	DECOMP/1	DECOMP/2	DECOMP/3	DECOMP/4
SCSD8	2	1.0	1.2	1.6	1.7
	4	1.0	1.3	1.8	2.0
	6	1.0	1.5	1.8	2.2
	8	1.0	1.6	1.6	2.2
	10	1.0	1.6	2.3	2.7
	12	1.0	1.6	2.5	2.5
	14	1.0	1.3	1.6	2.4
	16	1.0	2.3	3.2	3.9
	18	1.0	1.6	2.5	2.4
	20	1.0	1.9	2.6	4.2
	22	1.0	1.7	2.3	3.4
	24	1.0	1.6	2.8	3.1
	26	1.0	1.6	2.4	4.5
	28	1.0	1.9	3.7	4.1
	30	1.0	1.5	2.9	2.6
	32	1.0	2.2	2.7	4.7
	34	1.0	1.8	2.5	3.7
	36	1.0	1.8	2.1	3.1
	38	1.0	1.6	3.5	4.6

Table C.9: Speedup for a varying number of subproblems.

C.3 Varying the Number of Processors

The last table supports Figures 4.13 and 4.14, and following are descriptions of their column headings.

Problem Name The format is Name/N/p, where Name is the name of the test problem, N is the number of subproblems, and p is the number of processors.

Total Iterations The simplex method iterations done on all subproblems.

Degen Iterations The number of degenerate simplex iterations done on all subproblems.

Total Solves The number of subproblems solved.

Total Work The number of CPU seconds used for the entire run.

Power The effective number of CPUs applied to the problem: Work/Time.

Work The number of CPU seconds used to form and solve the subproblems.

Time The Elapsed seconds needed to form and solve the subproblems.

Obective Value The objective value obtained for the overall LP problem.

Problem Name	Total Itns.	Degen Itns.	Total Solves	Total Work	Power	Work	Time	Objective Value
SCSD8/39/1	14195	3952	2793	31.1	1.0	28.0	28.0	9.0499999992489E+02
SCSD8/39/2	17970	5100	3457	39.7	2.0	35.8	17.9	9.0499999998733E+02
SCSD8/39/3	14598	4234	2626	31.7	2.9	27.8	9.4	9.0500000005023E+02
SCSD8/39/4	16402	4849	3375	39.4	4.0	34.6	8.7	9.0499999989715E+02
SCSD8/39/5	14997	4624	3106	37.1	4.8	31.9	6.6	9.0499999993473E+02
SCSD8/39/6	9967	3647	1861	25.9	5.6	20.9	3.7	9.0499999991451E+02
SCSD8/39/7	16721	4988	3121	40.5	6.5	34.6	5.3	9.0499999992473E+02

Table C.10: Power, Work and Time for a varying number of processors.

Bibliography

[Abr83] Philip G. Abrahamson (1983). A Nested Decomposition Approach for Solving Staircase Linear Programs. Ph.D. Dissertation, Department of Operations Research, Stanford University, Stanford, California.

[ARS8] Byong-Hun Ahn and Seung-Kyu Rhee (1988). Cooperative variant of Dantzig-Wolfe decomposition method. Draft Report, Department of Management Science, Korea Advanced Institute of Science and Technology, Seoul, KOREA.

[Bea55] E. Martin L. Beale (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistics Society* 17, 173–184.

[Ben62] Jacques F. Benders (1962). Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathematik* 4, 238–252.

[BeT89] Dimitri P. Bertsekas and John Tsitsiklis (1989). *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ.

[Bir85] John R. Birge (1985). Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research* 33, 989–1007.

[Bun76] James R. Bunch (1976). Block methods for solving sparse linear systems. *Sparse Matrix Computations*, James R. Bunch and Donald J. Rose (editors). Academic Press, New York, New York.

[CDM79] Harlan Crowder, Ron S. Dembo and John M. Mulvey (1979). On reporting computational experiments with mathematical software. *ACM Transactions on Mathematical Software* 5, 193–203.

[Chv83] Vašek Chvátal (1983). *Linear Programming.* W. H. Freeman and Company, New York and San Francisco.

[Dan55] George B. Dantzig (1955). Linear programming under uncertainty. *Management Science* 1, 197–206.

[Dan59] George B. Dantzig (1955). On the status of multistage linear programming problems. *Management Science* 6, 53–72.

[Dan63] George B. Dantzig (1963). *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey.

[Dan73] George B. Dantzig (1973). Solving staircase linear programs by a nested block-angular method. Report SOL 73-1, Department of Operations Research, Stanford University, California.

[Dan87] George B. Dantzig (1987). Origins of the simplex method. Report SOL 87-5, Department of Operations Research, Stanford University, California.

[DF87] Jack J. Dongarra and Eric Grosse (1987). Distribution of mathematical software via electronic mail. *Communications of the ACM* 30, 403–407.

[DG88] George B. Dantzig and Peter W. Glynn (1988). Parallel processors for planning under uncertainty. Report SOL 88-8R, Department of Operations Research, Stanford University, California.

[DGG64] William S. Dorn, Ralph E. Gomory, and Harvey S. Greenberg (1964). Automatic design of optimal structures. *Journal de Mechanique* 3, 25–52.

[Duf84] Iain S. Duff (1984). Data structures, algorithms and software for sparse matrices. AERE Harwell Report CSS 158, HMSO, London.

[DW61] George B. Dantzig and Philip Wolfe (1961). The decomposition algorithm for linear programming. *Econometrica*, Vol. 29, No. 4.

[EWS8] Yuri Ermoliev and Roger J-B Wets (Editors) (1988). *Numerical Techniques of Stochastic Optimization*, Springer-Verlag, Berlin.

[Fou89] John J. H. Forrest (1989). Conversation on the IBM VMS operating system. IBM T. J. Watson Research Center, Yorktown Heights, New York.

[Fou82] Robert W. Fourer (1982). Solving staircase linear programs by the simplex method, 1: Inversion. *Mathematical Programming* 23, 274–313.

[FTS8] John J. H. Forrest and John A. Tomlin (1988). Vector processing in simplex and interior methods for linear programming. Presented at the workshop on *Supercomputers and Large Scale Optimization*, University of Minnesota, May 1988.

[Gay85] David M. Gay (1985). Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, December 1985.

[Geo70] Arthur M. Geoffrion (1970). Elements of large-scale mathematical programming: Parts I and II. *Management Science* 16, 652–691.

[GLS9] J. Alan George and Joseph W-H. Liu (1989). On the evolution of the minimum degree algorithm. *SIAM Review* 31, 1-19.

[Gla71] C. Roger Glassey (1971). Dynamic linear programs for production scheduling. *Operations Research* 19, 45–56.

[Gla73] C. Roger Glassey (1973). Nested decomposition and multi-stage linear programs. *Management Science* 20, 282–292.

[Gol56] Alan J. Goldman (1956). Resolution and separation theorems for polyhedral convex sets, in *Linear Inequalities and Related Systems,* Harold W. Kuhn and Albert W. Tucker (editors), Princeton University Press.

[Gol86] E. G. Gol'shteĭn (1986). The block method of convex programming. *Soviet Math. Dokl.* 33, 584-587.

[Gol87] E. G. Gol'shteĭn (1987). A general approach to decomposition of optimization systems. *Soviet Journal of Computing and Systems Science* 25, 105-114 [translated from *Tekhnicheskaya Kibernetika* 1, 59-69, (1987)].

[GMSW87] Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright (1987). Maintaining *LU* factors of a general sparse matrix. *Linear Algebra and its Applications* 88/89, 239-270.

[Hie82] Kathie L. Hiebert (1982). An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Transactions on Mathematical Software* 8, 5-20.

[Ho74] James K. Ho (1974). Nested decomposition of large scale linear programs with the staircase structure. Ph.D. Dissertation, Department of Operations Research, Stanford University, Stanford, California.

[HLS1a] James K. Ho and Etienne Loute (1981). A set of staircase linear programming test problems. *Mathematical Programming* 20, 245-250.

[HLS1b] James K. Ho and Etienne Loute (1981). An advanced implementation of the Dantzig-Wolfe decomposition algorithm for linear programming. *Mathematical Programming* 20, 303-326.

[HLS88] James K. Ho, Tak C. Lee and R. P. Sundarraj (1988). Decomposition of linear programs using parallel computation. *Mathematical Programming* 242, 391-405.

[JBNPS9] Richard H. F. Jackson, Paul T. Boggs, Stephen G. Nash and Susan Powell (1989). *Report of the ad hoc committee to revise the guidelines for reporting computational experiments in mathematical programming.* Mathematical Programming Society, Committee on Algorithms.

[Kal76] Peter Kall (1976). *Stochastic Programming.* Springer-Verlag, Berlin.

[LooS6a] Frcek A. Lootsma (1986). State-of-the-art in parallel unconstrained optimization. *Parallel Computing* 5, 157–163.

[LoRS8] Frcek A. Lootsma and Kenneth M. Ragsdell (1988). State-of-the-art in parallel nonlinear optimization. *Parallel Computing* 6, 131–155.

[LusS7] Irvin J. Lustig (1987). An analysis of an available set of linear programming test problems. Report SOL 87-11, Department of Operations Research, Stanford University, California.

[Mar57] Harold M. Markowitz (1957). The elimination form of the inverse and its application to linear programming. *Management Science,* 3, 255–269.

[MSS7] Bruce A. Murtagh and Michael A. Saunders (1987). MINOS 5.1 user's guide (revised). Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, California.

[ReiS6] John K. Reid (1986). Sparse matrices. AERE Harwell Report CSS 201, HMSO, London.

[Roc76] R. Tyrell Rockafellar (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization* 14, 877–898.

[Ros70] Donald J. Rose (1970). Symmetric elimination on sparse positive definite systems and potential flow network problems. Ph.D. Thesis, Harvard University, Cambridge, Massachusetts.

[RoyS3] Tony J. van Roy (1983). Cross decomposition for mixed integer programming. *Mathematical Programming* 25, 46-63.

[StuS8] Craig B. Stunkel (1988). Linear optimization via message-based parallel processing. Proceedings of the 1988 International Conference on Parallel Processing, Volume III, 264–271. The Pennsylvania State University Press, University Park, Pennsylvania.

[Tar76] Robert E. Tarjan (1976). Graph theory and Gaussian elimination. *Sparse Matrix Computations*, James R. Bunch and Donald J. Rose (editors). Academic Press, New York, New York.

[TomS7] John A. Tomlin (1987). Mathematical programming and supercomputers. Ketron Management Science, Inc., Mountain View, California.

[VW69] Richard Van Slyke and Roger J-B Wets (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics* 17, 638–663.

[VS64] Richard Van Slyke (1964). Mathematical Programming and Optimal Control. Ph.D. thesis, University of California, Berkeley.

[WelS9] Joseph Wells (1989). Conversation on the IBM MVS operating system. IBM T. J. Watson Research Center, Yorktown Heights, New York.

[WitS3] Robert J. Wittrock (1983). Advances in a Nested Decomposition Algorithm for Solving Staircase Linear Programs. Ph.D. Dissertation, Department of Operations Research, Stanford University, Stanford, California.

[Zad62] Lotfi Zadeh (1962). Note on linear programming and optimal control. *IRE transactions on Automatic Control* 7.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 89-17	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Parallel Decomposition of Linear Programs		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert Entriken		8. CONTRACT OR GRANT NUMBER(s) N00014-89-J-1659 Grant N00014-87-K-0142 Contract
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305-4022		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1111NA
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE November 1989 13. NUMBER OF PAGES 150 pp.
		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

linear program; optimization; decomposition; parallel computers; mathematical program

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(please see reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The Parallel Decomposition of Linear Programs
by Robert Entriken
SOL 89-17

Abstract

This thesis introduces a new calculus for manipulating linear-program decomposition schemes. A linear program is represented by a *communication network*, which is decomposed by splitting nodes in two, and a transformation is defined to recover subproblems from the network. We also define a dual-symmetric oracle that provides solutions to linear programs, and can be performed by the simplex method, nested decomposition, and finally, parallel decomposition.

Two important classes of linear program serve as examples for the above calculus: staircase linear programs and stochastic linear programs. For the former case, a sophisticated yet experimental computer code has been written for an IBM 3090/600E with six processors. The code performs the parallel decomposition algorithm and is tested on twenty-two small to medium sized "real-world" problems. Experiments show that in addition to speedups provided by decomposition alone, performance is improved by using parallel processors.